sc_lib

COLLABORATORS

	TITLE : sc_lib		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

sc_lib

1.1 sc_lib.guide

```
Common Problems

Alphabetical list of all C functions and data items

Character Type Macros and Functions
String Functions
Conversion Functions
Mathematical Macros and Functions
Varying-Length Argument List Macros
General Utility Macros and Functions
Sorting Functions
Random Number Generation Functions
Program Control Functions
Memory Allocation Functions
Memory Manipulation Functions
Diagnostic Control Functions
Time Functions
I/O Functions
Error Handling Functions
File Control Functions
File Positioning Functions
File Management Functions
System Interface Functions
Directory Functions
Localization Functions
Multibyte Character Functions
Screen Control Functions
Functions User Can Replace, But Not Call
Builtin Functions
C++ Classes
Useful Tables
Glossary
```

1.2 alphalist

1.3 alphalist abort() - __datecvt()

abort()	abs()	access()
acos()	argopt()	asctime()
asin()	assert()	astcsma()
atan()	atan2()	atexit()
atof()	atoi()	atol()
__autoopenfail()	_BackGroundIO	_Backstdout
bldmem()	bsearch()	__buffsize
calloc()	ceil()	chdir()
chgclk()	chkabort()	chkml()
chkufb()	chmod()	clearerr()
clock()	close()	closedir()
clrerr()	cos()	cosh()
cot()	creat()	ctime()
__ctype	_CXBRK()	_CXFERR()
_CXOVF()	datecmp()	__datecvt()

1.4 alphalist __daylight - fmod()

__daylight	__daytbl	_dclose()
_dcreat()	_dcreatx()	dfind()
difftime()	DiskfontBase	div()
dnext()	_dopen()	DOSBase
dos_packet()	dqsort()	drand48()
_dread()	_dseek()	_dwrite()
ecvt()	__emit()	_EPILOG()
erand48()	errno	except()
__exit()	exit()	exp()
fabs()	fclose()	fcloseall()
fcvt()	fdopen()	feof()
ferror()	fflush()	fgetc()
fgetchar()	fgetpos()	fgets()
fileno()	findpath()	floor()
flushall()	__fmask	fmod()

1.5 alphalist __fmode - intuitionbase

__fmode	fmode()	fopen()
forkl()	forkv()	_FPERR
fprintf()	fputc()	fputchar()
fputs()	fqsort()	fread()
free()	freopen()	frexp()
fscanf()	fseek()	fsetpos()
fstat()	ftell()	fwrite()
gcvt()	geta4()	getasn()
getc()	getcd()	getch()
getchar()	getclk()	getcwd()
getdfs()	getenv()	getfa()
getfnl()	getft()	getmem()
getml()	getpath()	getreg()
gets()	GfxBase	gmtime()
halloc()	iabs()	IntuitionBase

1.6 alphalist iomode() - memcmp()

iomode()	isalnum()	isalpha()
isascii()	isatty()	iscntrl()
iscsym()	iscsymf()	isdigit()
isgraph()	islower()	isprint()
ispunct()	isspace()	isupper()
isxdigit()	jrand48()	labs()
lcong48()	ldexp()	ldiv()
localeconv()	localtime()	log()
log10()	longjmp()	lqsort()
lrand48()	lsbrk()	lseek()
lstat()	__main()	main()
malloc()	MathBase	__matherr()
MathTransBase	max()	mblen()
mbstowcs()	mbtowc()	memccpy()
memchr()	_MemCleanup()	memcmp()

1.7 alphalist memcpy() - rawcon()

memcpy()	memmove()	memset()
_MemType	min()	mkdir()
mkstemp()	mktemp()	mktime()
modf()	__montbl	__months
movmem()	mrand48()	__msflag
_MSTEP	nrand48()	__nufbs
offsetof()	onbreak()	onexit()
open()	opendir()	_OSERR
__os_errlist	__oslibversion	__os_nerr
ovlyMgr()	perror()	poserr()
pow()	pow2()	printf()
__priority	__procname	_PROLOG()
putc()	putchar()	putenv()
putreg()	puts()	qsort()
raise()	rand()	rawcon()

1.8 alphalist rbrk() - sizmem()

rbrk()	read()	readdir()
readlocale()	realloc()	remove()
rename()	repmem()	rewind()
rewinddir()	rlsmem()	rlsml()
rmdir()	rstmem()	sbrk()
scanf()	scdir()	scr_beep()
scr_bs()	scr_cdelete()	scr_cinsert()
scr_clear()	scr_cr()	scr_curs()
scr_cursrt()	scr_cursup()	scr_eol()
scr_home()	scr_ldelete()	scr_lf()
scr_linsert()	scr_tab()	seed48()
seekdir()	setbuf()	setjmp()
setlocale()	setmem()	setnbf()
setvbuf()	__sigfunc()	signal()
sin()	sinh()	sizmem()

1.9 alphalist _slash - stpcpy()

_SLASH	sprintf()	sqrt()
sqsort()	srand()	srand48()
sscanf()	__stack	stackavail()
stacksize()	stackused()	stat()
stcarg()	stccpy()	stcd_i()
stcd_l()	stcgfe()	stcgfn()
stcgfp()	stch_i()	stch_l()
stcis()	stcisn()	stci_d()
stci_h()	stci_o()	stclen()
stcl_d()	stcl_h()	stcl_o()
stco_i()	stco_l()	stcpm()
stcpma()	stcsma()	stcul_d()
stcu_d()	__stdiov37	__stdiowin
__STKNEED	stpblk()	stpbrk()
stpchr()	stpchrn()	stpcpy()

1.10 alphalist stpdate() - _stub()

stpdate()	stpsym()	stptime()
stptok()	strbpl()	strcat()
strchr()	strcmp()	strcmpi()
strcoll()	strcpy()	strcspn()
strdup()	strerror()	strftime()
stricmp()	_STRICT_ANSI	strins()
strlen()	strlwr()	strmfe()
strmfn()	strmfp()	strmid()
strncat()	strncmp()	strncpy()
strnicmp()	strnset()	strpbrk()
strrchr()	strrev()	strset()
strsfn()	strspn()	strsrt()
strstr()	strtod()	strtok()
strtol()	strtoul()	strupr()

strxfrm()	stspfp()	_stub()

1.11 alphalist swmem() - _xcexit()

swmem()	SysBase	__sys_errlist
__sys_nerr	system()	tan()
tanh()	tell()	telldir()
time()	__timecvt()	timer()
__timezone	tmpfile()	tmpnam()
__tinymain()	toascii()	tolower()
toupper()	tqsort()	_TZ
__tzdtn	__tzname	__tzset()
__tzstn	__ufbs	ungetc()
unlink()	utpack()	utunpk()
va_arg()	va_end()	va_start()
vfprintf()	vprintf()	vsprintf()
wait()	waitm()	_WBenchMsg
wcstombs()	wctomb()	write()
_XCEXIT()		

1.12 charfuncs

Function	Portability	Description
isalnum	ANSI	check if a character is alphanumeric
isalpha	ANSI	check if a character is alphabetic
isascii	SAS/C	check if a character is an ASCII character
iscntrl	ANSI	check if a character is a control character
iscsym	SAS/C	check if a character is a C symbol character
iscsymf	SAS/C	check if a character is a C symbol lead character
isdigit	ANSI	check if a character is a decimal digit (0-9)
isgraph	ANSI	check if a character is a graphic character
islower	ANSI	check if a character is lowercase
isprint	ANSI	check if a character is printable
ispunct	ANSI	check if a character is a punctuation character
isspace	ANSI	check if a character is a space
isupper	ANSI	check if a character is uppercase
isxdigit	ANSI	check if a character is a hex character (0-9, A-F, a-f)
toascii	SAS/C	convert a character to an ASCII character
tolower	ANSI	convert a character to lowercase
toupper	ANSI	convert a character to uppercase

1.13 stringfuncs

See also "Memory Manipulation Functions," later in this section.

Function	Portability	Description
astcsma	AmigaDOS	AmigaDOS string pattern match (anchored)

scdir	OLD	Return the name of the next file matching pattern
stcpm	SAS/C	unanchored pattern match
stcpma	SAS/C	anchored pattern match
stcsma	AmigaDOS	UNIX string pattern match (anchored)
stccpy	UNIX	copy one string to another
stpcpy	SAS/C	copy one string to another
strcpy	ANSI	copy one string to another
strncpy	ANSI	copy a string, length limited
stcis	SAS/C	count the number of string characters in the set
stcisn	SAS/C	count the number of string characters not in the set
strcspn	ANSI	count the number of string characters not in the set
stclen	OLD	measure the length of a string
strlen	ANSI	measure the length of a string
strspn	ANSI	count longest span of characters in the set
stpbrk	OLD	find a break character in a string
stpchr	OLD	find a character in a string
strchr	ANSI	find a character in a string
stpchrn	OLD	find a character not in a string
strrchr	ANSI	find a character not in a string
strpbrk	ANSI	find a break character in a string
strcat	ANSI	concatenate strings
strncat	ANSI	concatenate strings, length limited
strcmp	ANSI	compare strings, case sensitive
strcmpi	OLD	compare strings, case insensitive
stricmp	SAS/C	compare strings, case insensitive
strncmp	ANSI	compare strings, length limited
strnicmp	SAS/C	compare strings, case insensitive, length limited
strnset	XENIX	set a string to a value, length limited
strset	XENIX	set a string to a value
strtok	ANSI	get a token
stcarg	SAS/C	get an argument
stpsym	SAS/C	get the next symbol from a string
stptok	SAS/C	get the next token from a string
stpblk	SAS/C	skip blanks
strbpl	SAS/C	build a string-pointer list
strdup	XENIX	duplicate a string
strins	SAS/C	insert a string
strrev	XENIX	reverse a character string
strmid	SAS/C	return a substring from a string
strstr	ANSI	find a substring inside of a string

1.14 convfuncs

Function	Portability	Description
atof	ANSI	convert an ASCII string to a floating-point number
atoi	ANSI	convert an ASCII string to an integer
atol	ANSI	convert an ASCII string to a long integer
stcd_i	SAS/C	convert a decimal string to an integer
stcd_l	SAS/C	convert a decimal string to a long integer
ecvt	UNIX	convert a double-precision floating-point number to a string

fcvt	UNIX	convert a floating-point number to a string
gcvt	UNIX	convert a floating-point number to a string
stch_i	SAS/C	convert a hexadecimal string to an integer
stch_l	SAS/C	convert a hexadecimal string to a long integer
stci_d	SAS/C	convert an integer to a decimal string
stci_h	SAS/C	convert an integer to a hexadecimal string
stci_o	SAS/C	convert an integer to an octal string
stcl_d	SAS/C	convert a long integer to a decimal string
stcl_h	SAS/C	convert a long integer to a hexadecimal string
stcl_o	SAS/C	convert a long integer to an octal string
stco_i	SAS/C	convert an octal string to an integer
stco_l	SAS/C	convert an octal string to a long integer
stcu_d	SAS/C	convert an unsigned integer to a decimal string
stcul_d	SAS/C	convert an unsigned long integer to a decimal string
strlwr	XENIX	convert a string to lowercase
strtod	ANSI	convert a string to a double-precision floating-point number
strtol	ANSI	convert a string to a long integer
strtoul	ANSI	convert a string to an unsigned long integer
toascii	SAS/C	convert a character to an ASCII string
tolower	ANSI	convert a character to lowercase
toupper	ANSI	convert a character to uppercase
mbstowcs	ANSI	convert a multibyte string to a wide-character string
wcstombs	ANSI	convert a wide-character string to a multibyte string
stpdate	SAS/C	convert a date array to a string
stptime	SAS/C	convert a time array to a string
mktime	ANSI	convert a broken down time to a time_t value
__datecvt	AmigaDOS	convert an AmigaDOS DateStamp to a time_t value
__timecvt	AmigaDOS	convert a time_t value to an AmigaDOS DateStamp
ctime	ANSI	convert a time_t value to an ASCII string
utpack	SAS/C	pack UNIX time
utunpk	SAS/C	unpack UNIX time

1.15 mathfuncs

Function	Portability	Description
abs	ANSI	absolute value
acos	ANSI	arccosine function
asin	ANSI	arcsine function
atan	ANSI	arctangent function
atan2	ANSI	arctangent of x/y
ceil	ANSI	get floating-point limits
cos	ANSI	cosine function
cosh	ANSI	hyperbolic cosine function
cot	UNIX	cotangent function
div	ANSI	compute the quotient and the remainder
exp	ANSI	exponential function
fabs	ANSI	floating-point or double-precision floating-point absolute value
floor	ANSI	get the floor of a real number
fmod	ANSI	floating-point modulus operations

frexp	ANSI	split floating-point value
iabs	SAS/C	integer absolute value
labs	ANSI	long integer absolute value
ldexp	ANSI	combine floating-point value
ldiv	ANSI	return the long integer quotient and the remainder
log	ANSI	natural logarithm function
log10	ANSI	base 10 logarithm function
max	UNIX	compute the maximum of two values
min	UNIX	compute the minimum of two values
modf	ANSI	split a floating-point value
pow	ANSI	raise a number to a power
pow2	SAS/C	raise 2 to a power
sin	ANSI	sine function
sinh	ANSI	hyperbolic sine function
sqrt	ANSI	square root function
tan	ANSI	tangent function
tanh	ANSI	hyperbolic tangent function

1.16 varargfuncs

Function	Portability	Description
va_arg	ANSI	get an argument from a varying-length argument list
va_end	ANSI	end varying-length argument list processing
va_start	ANSI	begin varying-length argument list processing

1.17 generalfuncs

Function	Portability	Description
__emit	SAS/C	emit 680x0 instruction word
getreg	SAS/C	get 680x0-specific registers
putreg	SAS/C	set up 680x0-specific registers
geta4	OLD	establish addressability to the global data area
isatty	UNIX	test a file descriptor for a terminal device
ovlyMgr	AmigaDOS	overlay manager call point
offsetof	ANSI	get the byte offset of a structure member

1.18 sortfuncs

Function	Portability	Description
bsearch	ANSI	perform a binary search
dqsort	SAS/C	sort an array of double-precision floating-point numbers
fqsort	SAS/C	sort an array of floating-point numbers
lqsort	SAS/C	sort an array of long integers
qsort	ANSI	sort a data array

sqsort	SAS/C	sort an array of short integers
strsrt	SAS/C	sort a string-pointer list
tqsort	SAS/C	sort an array of text pointers

1.19 randfuncs

Function	Portability	Description
drand48	UNIX	generate a random double-precision floating-point number (internal seed)
erand48	UNIX	generate a random double-precision floating-point number (external seed)
jrand48	UNIX	generate a random long integer (external seed)
lcong48	UNIX	set linear congruence parameters
lrand48	UNIX	generate a random positive long integer (internal seed)
mrand48	UNIX	generate a random long integer (internal seed)
nrand48	UNIX	generate a random positive long integer (external seed)
rand	ANSI	generate a random number
seed48	UNIX	set all 48 bits of an internal seed
srand	ANSI	set a seed for the rand function
srand48	UNIX	set high 32 bits of an internal seed

1.20 progfuncs

Function	Portability	Description
abort	ANSI	abort the current process
atexit	ANSI	set an exit trap
__autoopenfail	AmigaDOS	terminate the program if OpenLibrary fails
chkabort	AmigaDOS	check for a break character
Chk_Abort	AmigaDOS	check for a break character
_CXBRK	AmigaDOS	print message and exit for Control-C processing
exit	ANSI	terminate program execution
__exit	SAS/C	terminate program execution with no clean-up
onexit	OLD	set an exit trap
_XCEXIT	SAS/C	terminate the program
forkl	SAS/C	create a child process with an argument list
forkv	SAS/C	create a child process with an argument vector
longjmp	ANSI	perform a long jump
setjmp	ANSI	set long jump parameters
onbreak	AmigaDOS	plant a break trap
wait	SAS/C	wait for a single child process to complete
waitm	SAS/C	wait for multiple child processes to complete
raise	ANSI	generate a signal
signal	ANSI	establish event traps

1.21 allocfuncs

Function	Portability	Description
bldmem	OLD	build a memory pool of a specified size
rstmem	OLD	reset a memory pool
sizmem	SAS/C	get the memory pool size
chkml	SAS/C	check for the largest memory block
getmem	OLD	get a memory block (short)
getml	OLD	get a memory block (long)
halloc	SAS/C	allocate a huge memory block
lsbrk	OLD	allocate memory
sbrk	OLD	allocate memory (unsigned)
calloc	ANSI	allocate and clear memory
malloc	ANSI	allocate memory
realloc	ANSI	reallocate memory
free	ANSI	free memory
_MemCleanup	AmigaDOS	deallocate all allocated memory
rbrk	OLD	release memory
rlsmem	OLD	release a memory block
rlsml	OLD	release a memory block

1.22 memoryfuncs

Function	Portability	Description
memchr	ANSI	find a character in a memory block
memcmp	ANSI	compare two memory blocks
memccpy	SAS/C	copy a memory block
memcpy	ANSI	copy a memory block
memmove	ANSI	copy bytes in memory
memset	ANSI	set a memory block to a value
movmem	UNIX	move a memory block
repmem	SAS/C	replicate values through a block
setmem	UNIX	set a memory block to a value
swmem	UNIX	swap two memory blocks

1.23 diagfuncs

Function	Portability	Description
assert	ANSI	assert program validity
except	SAS/C	call the math error handler
__matherr	UNIX	math error handler
perror	ANSI	print an error message
poserr	AmigaDOS	print an AmigaDOS error message
strerror	ANSI	print the text for a given error number

1.24 timefuncs

Function	Portability	Description
asctime	ANSI	generate an ASCII time string
clock	ANSI	measure program processor time
datecmp	AmigaDOS	compare two AmigaDOS dates
difftime	ANSI	compute the difference between two time_t values
gmtime	ANSI	unpack Greenwich Mean Time (GMT)
localtime	ANSI	unpack local time
mktime	ANSI	convert a broken down time to a time_t value
strftime	ANSI	format a time string
time	ANSI	get the system time in seconds
ctime	ANSI	convert a time_t value to an ASCII string
__timecvt	AmigaDOS	convert a time_t value to an AmigaDOS DateStamp
timer	AmigaDOS	get the system clock with microseconds
__tzset	XENIX	set the time zone variables
utpack	SAS/C	pack UNIX time
utunpk	SAS/C	unpack UNIX time

1.25 iofuncs

```
     Standard I/O Functions
     Character I/O Functions
     String I/O Functions
     Block I/O Functions
     Formatted I/O Functions
```

1.26 standardiofuncs

Function	Portability	Description
getch	SAS/C	get a character from stdin immediately
getchar	ANSI	get a character from stdin
fgetchar	XENIX	get a character from stdin
fputchar	XENIX	put a character to stdout
putchar	ANSI	put a character to stdout
gets	ANSI	get a string from stdin
puts	ANSI	put a string to stdout
printf	ANSI	formatted print to stdout
scanf	ANSI	formatted input conversions
vprintf	ANSI	formatted write of a varying-length argument list to stdout

1.27 characteriofuncs

Function	Portability	Description
fgetc	ANSI	get a character from a file
getc	ANSI	get a character from a file
ungetc	ANSI	push an input character back

getch	SAS/C	get a character from stdin immediately
getchar	ANSI	get a character from stdin
fgetchar	XENIX	get a character from stdin
fputc	ANSI	put a character to a level 2 file
fputchar	XENIX	put a character to stdout
putc	ANSI	put a character to a level 2 file
putchar	ANSI	put a character to stdout

1.28 stringiofuncs

Function	Portability	Description
fgets	ANSI	get a string from a level 2 file
gets	ANSI	get a string from stdin
fputs	ANSI	put a string to a level 2 file
puts	ANSI	put a string to stdout

1.29 blockiofuncs

Function	Portability	Description
_dread	OLD	read an AmigaDOS file
_dwrite	OLD	write to an AmigaDOS file
fread	ANSI	read and write blocks
fwrite	ANSI	write blocks to a level 2 file
read	UNIX	read from a level 1 file
write	UNIX	write to level 1 file

1.30 formattediofuncs

Function	Portability	Description
fprintf	ANSI	formatted print
printf	ANSI	formatted print to stdout
sprintf	ANSI	formatted print to a string
fscanf	ANSI	formatted input conversions
scanf	ANSI	formatted input conversions
sscanf	ANSI	formatted input conversions
vfprintf	ANSI	formatted write of a varying-length argument list to a file
vprintf	ANSI	formatted write of a varying-length argument list to stdout
vsprintf	ANSI	formatted write of a varying-length argument list to a string

1.31 errorfuncs

Function	Portability	Description
feof	ANSI	check for a level 2 end-of-file
ferror	ANSI	check for a level 2 error
clearerr	ANSI	clear a level 2 I/O error flag
clrerr	UNIX	clear a level 2 I/O error flag

1.32 filectlfuncs

Function	Portability	Description
close	UNIX	close a level 1 file
_dclose	OLD	close an AmigaDOS file
fclose	ANSI	close a level 2 file
fcloseall	XENIX	close all level 2 files
tmpnam	ANSI	create a temporary filename
creat	UNIX	create a level 1 file
_dcreat	OLD	create an AmigaDOS file
_dcreatx	OLD	create a new AmigaDOS file
fdopen	UNIX	attach a level 1 file to level 2
fileno	UNIX	get the file number for a level 2 file
fmode	SAS/C	change the mode of a level 2 file
iomode	SAS/C	change the mode of a level 1 file
open	UNIX	open a level 1 file
_dopen	OLD	open an AmigaDOS file
fopen	ANSI	open a level 2 file
tmpfile	ANSI	open a temporary file stream
freopen	ANSI	reopen a level 2 file
fflush	ANSI	flush a level 2 output buffer
flushall	XENIX	flush all level 2 output buffers
mkstemp	UNIX	Make a unique filename and open the file
mktemp	UNIX	Make a unique filename
setbuf	ANSI	set the buffer mode for a level 2 file
setnbf	UNIX	turn off I/O buffering for a level 2 file
setvbuf	ANSI	set the variable buffer for a level 2 file

1.33 fileposfuncs

Function	Portability	Description
_dseek	OLD	reposition an AmigaDOS file
fseek	ANSI	set a level 2 file position
lseek	UNIX	set a level 1 file position
fgetpos	ANSI	get the current file position
fsetpos	ANSI	reposition a file
ftell	ANSI	get a level 2 file position
tell	UNIX	get a level 1 file position
rewind	ANSI	reset a level 2 file position to its first byte

1.34 filemanfuncs

Function	Portability	Description
access	UNIX	check file accessibility
chkufb	SAS/C	check a level 1 file handle
fileno	UNIX	get the file number for a level 2 file
chmod	UNIX	change a file’s protection mode
fmode	SAS/C	change the mode of a level 2 file
iomode	SAS/C	change the mode of a level 1 file
fstat	UNIX	get file status
getfa	AmigaDOS	get the file attribute
getft	AmigaDOS	get the file time
stat	UNIX	get information on a file
stcgfe	SAS/C	get the filename extension
stcgfn	SAS/C	get a filename from a path
stcgfp	SAS/C	get a file path
strmfe	SAS/C	make a filename with an extension
strmfn	SAS/C	make a filename from components
strmfp	SAS/C	make a filename from path or node
strsfn	SAS/C	split the filename
unlink	UNIX	remove a file
remove	ANSI	remove a file
rename	ANSI	rename a file
setbuf	ANSI	set the buffer mode for a level 2 file
setnbf	UNIX	turn off I/O buffering for a level 2 file
setvbuf	ANSI	set the buffer mode for a level 2 file

1.35 sysfuncs

Function	Portability	Description
argopt	SAS/C	get options from an argument list
chgclk	AmigaDOS	change the system clock
dos_packet	OLD	send AmigaDOS output packet
getclk	AmigaDOS	get or change the system clock
getasn	AmigaDOS	get assigned device
getdfs	AmigaDOS	get disk free space
getenv	ANSI	get environment variable
putenv	UNIX	put a string into the environment
rawcon	AmigaDOS	set or unset raw console input
stackavail	SAS/C	get the current available stack size
stacksize	SAS/C	get the current stack size
stackused	SAS/C	get the amount of stack in use
system	ANSI	call the system command processor

1.36 dirfuncs

Function	Portability	Description
chdir	UNIX	change the current directory
closedir	UNIX	terminate the directory operation

dfind	AmigaDOS	find a directory entry
dnext	AmigaDOS	find the next directory entry
findpath	AmigaDOS	locate a file in the current path
getcd	AmigaDOS	get the current directory
getcwd	UNIX	get the current working directory
getfnl	SAS/C	get a filename list
getpath	AmigaDOS	get the path for a specific directory or file
mkdir	UNIX	make a new directory
opendir	UNIX	initiate a directory operation
readdir	UNIX	read a directory element
rmdir	UNIX	remove a directory
seekdir	UNIX	reposition a directory operation
rewinddir	UNIX	reset to the start of the directory
telldir	UNIX	get the directory position

1.37 localfuncs

Function	Portability	Description
localeconv	ANSI	return information on locale formatting conventions
readlocale	SAS/C	read and initialize a new locale
setlocale	ANSI	set locale information for a program
strcoll	ANSI	compare strings based on locale
strxfrm	ANSI	transform a string

1.38 multifuncs

Function	Portability	Description
mblen	ANSI	determine the length of a multibyte character
mbstowcs	ANSI	convert a multibyte string to a wide-character string
mbtowc	ANSI	map a multibyte character to a wide character
wcstombs	ANSI	convert a wide-character string to a multibyte string
wctomb	ANSI	map a wide character to a multibyte character

1.39 screenfuncs

Function	Portability	Description
scr_beep	OLD	call Intuition DisplayBeep
scr_bs	OLD	move the cursor left one position
scr_cdelete	OLD	delete the character under the cursor
scr_cinsert	OLD	insert a blank character at the cursor
scr_clear	OLD	clear the window
scr_cr	OLD	cause a carriage return
scr_curs	OLD	move the cursor to specified line and column
scr_cursrt	OLD	move the cursor right one character

scr_cursup	OLD	move the cursor up one line
scr_eol	OLD	erase from the cursor to the end-of-line
scr_home	OLD	move the cursor to the upper left
scr_ldelete	OLD	delete the line at the cursor position
scr_lf	OLD	move the cursor down one line
scr_linsert	OLD	insert a blank line at the cursor
scr_tab	OLD	move the cursor right one tab stop

1.40 nocallfuncs

Function	Portability	Description
__autoopenfail	AmigaDOS	terminate the program if OpenLibrary fails
_CXFERR	SAS/C	low-level floating-point error
_CXOVF	SAS/C	default stack-overflow handler
_EPILOG	SAS/C	profiler function exit hook
_PROLOG	SAS/C	profiler function entry hook

1.41 bltinfuncs

Function	Portability	Description
strcpy	ANSI	copy one string to another
strlen	ANSI	measure the length of a string
strcmp	ANSI	compare strings
memcmp	ANSI	compare two memory blocks
memcpy	ANSI	copy a memory block
memset	ANSI	set a memory block to a value
abs	ANSI	absolute value
max	UNIX	compute the maximum of two values
min	UNIX	compute the miminum of two values
__emit	SAS/C	emit 680x0 instruction word
getreg	SAS/C	get 680x0-specific registers
putreg	SAS/C	set up 680x0-specific registers
geta4	OLD	establish addressability to the global data area

1.42 usefultables

C Operator Precedence
Formatted Input Specifiers
Formatted Output Specifiers

1.43 _backgroundio

_BackGroundIO-Route I/O to the console window

Synopsis

```
  #include <dos.h>

  long _BackGroundIO;
```

Description

The global variable _BackGroundIO is used by cback.o to indicate whether output will be sent to the console window by your background process. The default is 0, indicating that I/O will not be sent to the console window. Setting _BackGroundIO to 1 tells cback.o to initialize _Backstdout to point to the console window. If _BackGroundIO is set to 1, you must call Close on _Backstdout before the program exits:

```
    Close (_Backstdout);
```

If you do not include this statement before your program exits, the CLI window pointed to by _Backstdout will not close properly.

This variable is of interest only for background processes. If you do not link with cback.o, this variable is ignored.

Portability

SAS/C

See Also

_Backstdout

1.44 _backstdout

_Backstdout-Pointer to the console window file handle

Synopsis

```
  #include <dos.h>

  BPTR _Backstdout;
```

Description

The global variable _Backstdout is used by cback.o to point to the console window where I/O is to be routed. _Backstdout is an AmigaDOS file handle, like the ones returned from the AmigaDOS function Open. If _BackGroundIO is set to 1, cback.o initializes _Backstdout to point to the console window. Otherwise, _Backstdout is not initialized. If _BackGroundIO is set to 1 and _Backstdout is initialized, you must call Close for _Backstdout before the program exits.

```
    Close (_Backstdout);
```

If you do not include this statement before your program exits,

```
  the Shell window pointed to by _Backstdout will not close
  properly.

  This variable is of interest only for background processes.  If
  you do not link with cback.o, this variable is ignored.
```

Portability

   SAS/C

See Also

   _BackGroundIO

1.45 __buffsize

__buffsize-Level 2 I/O buffer size

Synopsis

```
  extern int __buffsize;
```

Description

```
  This external integer is used by the level 2 I/O system to specify
  the size of the buffers allocated for level 2 files.  This
  location is also used to specify the size of a buffer attached to
  a file with the  setbuf  or  setvbuf  function.  __buffsize must be
  set to the size of the buffer before the  setbuf  or  setvbuf
  function is called or before the first I/O operation; otherwise,
  the default buffer size is used.

  The buffer is not allocated when the file is opened.  Instead, the
  first I/O operation causes the buffer to be allocated from the
  local memory pool if one has not been previously specified with
  the  setbuf  or  setvbuf  function.  This means that if __buffsize is
  changed between the  fopen  call and the first I/O operation, the
  size of the buffer allocated for the file will be the value of
  __buffsize at the time of the I/O operation, not the value when
  the file was opened.

  In Version 5.10 and earlier, this variable was named _bufsiz.
```

Portability

   SAS/C

See Also

   fopen , setbuf

1.46 __ctype

__ctype-Character class table

Synopsis

```
  #include <ctype.h>
```

Description

The external character array __ctype is a table indicating the attributes of all ASCII characters. It is 257 bytes long; one byte per ASCII character plus one to handle the EOF character. The individual bits within the byte are set to indicate attributes as described below.

Bit	Value	Meaning
_U	1	uppercase alphabetic character (A-Z)
_L	2	lowercase alphabetic character (a-z)
_N	4	decimal digit (0-9)
_S	8	white space
_P	16	punctuation character
_C	32	control character
_B	64	blank
_X	128	hexadecimal digit (0-9, A-F, a-f)

All indexes are offset by one so that the table also handles the end-of-file character (-1). You must index the table by the character plus one.

Normally, you will not need to access the array directly, but will instead use the is... character test macros (isupper , isascii , and others) to access it for you.

In Version 5.10 and earlier, this variable was named _ctype.

Portability

SAS/C

Example

```
  #include <ctype.h>

  /* Modify the ctype array to treat the character 0x80 as    */
  /* white space.  Change 0x81 to be an uppercase character.  */
  /* This means isspace() will return TRUE on 0x80, and       */
  /* isupper(), isalpha(), isalnum(), isprint(), and isgraph()*/
  /* will return TRUE on 0x81.            */

  void main(void)
  {
      /* Make 0x80 a white space character */
      __ctype[0x80+1] |= _S;

      /* Make 0x81 an uppercase character */
      __ctype[0x81+1] |= _U;
  }
```

See Also

```
   isalnum ,  isalpha ,  isascii ,  iscntrl ,  isdigit ,  isgraph ,
   islower ,  isprint ,  ispunct ,  isspace ,  isupper ,  isxdigit
```

1.47 __daylight

__daylight-Daylight savings time flag

Synopsis

```
   extern int __daylight;
```

Description

```
  __daylight indicates whether daylight savings time is currently in
  effect.  This variable is initialized by the  __tzset  function
  and used by the  localtime  function to adjust from Greenwich Mean
  Time (GMT) to the local time.
```

Portability

```
   UNIX
```

See Also

```
   localtime ,  __timezone ,  __tzname ,
   __tzdtn ,  __tzset ,  __tzstn
```

1.48 __daytbl

__daytbl-Array of days by name

Synopsis

```
  extern char *__daytbl[];
```

Description

```
  The external array __daytbl contains seven pointers to character
  strings representing the days of the week.  Each string is three
  bytes long, plus a null terminator.  The values are as follows:

      Index String Meaning
      ----- ------ -------
        0     Sun   Sunday
        1     Mon   Monday
        2     Tue   Tuesday
        3     Wed   Wednesday
        4     Thu   Thursday
        5    Fri   Friday
        6    Sat   Saturday
```

Portability

```
   SAS/C
```

Example

```
  /* Replace the day table with a French version.  You   */
  /* might do this as a static DATA intialization if you */
  /* know you will be in French; or, you could call this */
  /* routine if and when the user selects French from a  */
  /* menu item somewhere.              */

  extern char *__daytbl[];

  void main(void)
  {
      __daytbl[0] = "Dim";  /* Dimanche (Sunday)    */
      __daytbl[1] = "Lun";  /* Lundi    (Monday)    */
      __daytbl[2] = "Mar";  /* Mardi    (Tuesday    */
      __daytbl[3] = "Mer";  /* Mercredi (Wednesday) */
      __daytbl[4] = "Jeu";  /* Jeudi    (Thursday)  */
      __daytbl[5] = "Ven",  /* Vendredi (Friday)    */
      __daytbl[6] = "Sam";  /* Samedi   (Saturday)  */
  }
```

See Also

```
   __montbl ,  __months
```

1.49 diskfontbase

DiskfontBase-Disk font library vector

Synopsis

```
  extern struct library *DiskfontBase;

  DiskfontBase = OpenLibrary("diskfont.library",«revision»);
```

Description

```
  This external location is used by various Amiga library routines
  that interface with the ROM Kernel text functions that deal with
  new fonts. It must be initialized by an OpenLibrary call before
  any of the disk font functions documented in the Amiga ROM Kernel
  manuals can be called.  It is expected to contain the base address
  of the disk font library vector table.

  You must close the library before your program terminates.

  For information about the «revision» parameter, see "Defining
  Library Bases" in Chapter 2, "Using the Commodore Libraries."
  NOTE:  Do not capitalize the letter 'f' in DiskfontBase.
```

Portability

AmigaDOS

1.50 dosbase

DOSBase-AmigaDOS library vector

Synopsis

```
  extern struct DosLibrary *DOSBase;

  DOSBase = OpenLibrary("dos.library",«revision»);
```

Description

This external location is used by various Amiga library routines that interface with AmigaDOS system functions. It is initialized by an OpenLibrary call in the startup code and is expected to contain the base address of the AmigaDOS system library vector table. If you do not link with the startup module c.o, and you make calls to AmigaDOS system functions or use SAS/C features that call AmigaDOS system functions, then you must first initialize this location by calling the OpenLibrary function.

You must close the library before your program terminates.

For information about the «revision» parameter, see "Defining Library Bases" in Chapter 2, "Using the Commodore Libraries."

Portability

AmigaDOS

Example

```
  /* This example demonstrates how to use OpenLibrary on */
  /* dos.library to initialize DOSBase.  It finishes by  */
  /* closing dos.library, as all programs which open it  */
  /* must do before terminating.             */

  #include <stdio.h>
  #include <stdlib.h>

  extern struct DosLibrary *DOSBase;

  void main(void)
  {
      /*  Revision number for AmigaDOS version 1.3  */
      int ver = 34;

      DOSBase = (struct DosLibrary *)OpenLibrary("dos.library",ver);
      /*  Make sure library opened OK */
      if (DOSBase == NULL)
    exit(EXIT_FAILURE);

      /*  Insert your code using DOSBase here ...  */

      CloseLibrary((struct Library *)DOSBase);
  }
```

1.51 errno

```
errno-UNIX error number

Synopsis

  #include <error.h>

  extern int errno;

Description

  The external integer errno is initialized to 0 at start-up time.
  Then, if an error is detected by one of the standard library
  functions, a nonzero value is placed there.  The standard library
  never resets errno.

  Programmers typically use this information in two ways.  In some
  cases, it is appropriate to check errno after a sequence of
  operations and abort if any error occurred along the way.  In
  other cases, errno is checked periodically, and if it is nonzero,
  the appropriate corrective action is taken.  Then, the application
  program resets errno before beginning the next processing phase.

  The  __sys_nerr  and  __sys_errlist  items are defined in a C source
  file named syserr.c and are used by the  perror  function to print
  messages that correspond to the code found in errno.

  Even though error information is normally placed into errno by the
  standard library functions, application programs can also use this
  technique to indicate problems.  However, you should be careful
  about adding new codes and messages just above the highest UNIX
  code currently defined, since new UNIX codes are occasionally
  added.  Also, it is recommended that you add application-dependent
  codes by extending the header file error.h, which contains
  symbolic definitions of the code numbers.  The currently defined
  symbols are as follows:

      Symbol  Meaning
      ------  -------
      EOSERR  operating system error
      EPERM user is not owner
      ENOENT  no such file or directory
      ESRCH no such process
      EINTR interrupted system call
      EIO   I/O error
      ENXIO no such device or address
      E2BIG arg list is too long
      ENOEXEC EXEC format error
      EBADF bad file number
      ECHILD  no child process
      EAGAIN  no more processes allowed
      ENOMEM  no memory available
      EACCES  access denied
      EFAULT  bad address
      ENOTBLK block device required
      EBUSY resource is busy
      EEXIST  file already exists
      EXDEV cross-device link
      ENODEV  no such device
      ENOTDIR is not a directory
      EISDIR  is a directory
      EINVAL  invalid argument
      ENFILE  no more files (units) allowed
      EMFILE  no more files (units) allowed for this process
      ENOTTY  not a terminal
      ETXTBSY text file is busy
      EFBIG file is too large
      ENOSPC  no space left
      ESPIPE  seek issued to pipe
      EROFS read-only file system
      EMLINK  too many links
      EPIPE broken pipe
      EDOM  math function argument error
      ERANGE  math function result is out of range
```

Portability

ANSI

See Also

_FPERR , perror

1.52 __fmask

__fmask-Default protection bits for opening a file

Synopsis

```
  extern unsigned long __fmask;
```

Description

This external integer
indicates the default protection for any file created
by the SAS/C file I/O routines, including all ANSI I/O routines.
This
flag has no effect on AmigaDOS I/O routines such as Open.
The default protection is read, write, and delete.

```
  You can modify __fmask. To reset __fmask,
  use the following values (defined in the fcntl.h file):

      Value Meaning
      ----- -------
      S_ISCRIPT script
      S_IPURE pure
      S_IARCHIVE  archive
      S_IREAD read
      S_IWRITE  write
      S_IEXECUTE  execute
      S_IDELETE delete
```

Portability

   SAS/C

See Also

   creat , fopen , open

1.53 __fmode

__fmode-Default level 2 I/O mode

Synopsis

```
  extern int __fmode;
```

Description

  This external integer is used by the fopen function to determine
  the translation mode to use when the programmer does not specify a
  mode in the fopen call.

  For AmigaDOS, it is set to O_RAW, which specifies untranslated
  mode.

Portability

   SAS/C

See Also

   fopen

1.54 _fperr

_FPERR-Floating-point error code

Synopsis

```
  #include <math.h>

Description

  _FPERR contains a nonzero value after any low-level floating-point
  operation encounters an error.  Low-level operations include
  addition, subtraction, multiplication, division, comparison, and
  conversion from one number representation to another (for example,
  floating point to double-precision floating point).

  The math.h file contains the declaration extern int _FPERR, so you
  do not need to include this statement in your program.

  The following table lists the error codes and their corresponding
  symbols from the math.h file:

      Symbol    Value   Meaning
      ------    -----   -------
      _FPEUND 1     underflow
      _FPEOVF 2     overflow
      _FPEZDV 3     divide by zero
      _FPENAN 4     not a valid number
      _FPECOM 5     not comparable

  When the error occurs, the low-level operation passes the
  appropriate error code to the  _CXFERR  function, which must store
  the code in _FPERR.  _FPERR is never reset by any low-level
  operation.

  For compatibility with previous releases, the error code values
  without the leading underscore are also defined in the math.h file
  unless you define the  _STRICT_ANSI  flag before including math.h.

  NOTE: This variable is only set when using the standard math
  libraries scm.lib and scms.lib.

Portability

   SAS/C

Example

  /*
   *  This example uses the division operation to produce
   *  floating point errors.  For example, Enter 0 for the
   *  divisor and 1 for the dividend to produce
   *  _FPERR = 3 (Divide by zero).
   */

  #include <math.h>
  #include <stdio.h>

  void main(void)
  {
      double a,b,c;
      while(feof(stdin) == 0)
      {
    printf("Enter divisor:  ");
    if (scanf("%lf",&a) != 1)
        exit(0);

    printf("Enter dividend: ");
    if (scanf("%lf",&b) != 1)
        exit(0);

    _FPERR = 0;

    c = b / a;
    printf("_FPERR = %d\n",_FPERR);
    printf("%lf / %lf = %lf\n\n",b,a,c);
      }
  }
```

See Also

   errno

1.55 gfxbase

GfxBase-Graphics library vector

Synopsis

```
  extern struct GfxBase *GfxBase;

  GfxBase = OpenLibrary("graphics.library",«revision»);
```

Description

This external location is used by various Amiga library routines that interface with the ROM Kernel graphics system functions. It must be initialized by an OpenLibrary call before any of the graphics functions documented in the Amiga ROM Kernel manuals can be called. It is expected to contain the base address of the graphics library vector table.

You must close the library before your program terminates.

For information about the «revision» parameter, see "Defining Library Bases" in Chapter 2, "Using the Commodore Libraries."

Portability

   AmigaDOS

Example

```
  /* This example demonstrates how to use OpenLibrary on  */
  /* graphics.library to initialize GfxBase.  It finishes */
  /* by closing graphics.library, as all programs   */
  /* which open it must do before terminating.    */

  #include <stdio.h>
  #include <stdlib.h>

  extern struct GfxBase *GfxBase;

  void main(void)
  {
      /* Revision number for AmigaDOS version 1.3 */
      int ver = 34;

      GfxBase = (struct GfxBase *)OpenLibrary("graphics.library",ver);

      /* Make sure library opened OK */
      if (GfxBase == NULL)
    exit(EXIT_FAILURE);

      /* Insert your code using GfxBase here ... */

      CloseLibrary((struct Library *)GfxBase);
  }
```

1.56 intuitionbase

IntuitionBase-Intuition library vector

Synopsis

```
  extern struct IntuitionBase *IntuitionBase;

  IntuitionBase = OpenLibrary("intuition.library",«revision»);
```

Description

This external location is used by various Amiga library routines that interface with the Intuition system functions. It must be initialized by an OpenLibrary call before any of the functions documented in the Amiga Intuition manuals can be called. It is expected to contain the base address of the Intuition library vector table.

You must close the library before your program terminates.

For information about the «revision» parameter, see "Defining Library Bases" in Chapter 2, "Using the Commodore Libraries."

Portability

AmigaDOS

Example

```
  /* This example demonstrates how to use OpenLibrary on */
  /* intuition.library to initialize IntuitionBase.  It  */
```

```
/* finishes by closing intuition.library, as all       */
/* programs which open it must do before terminating.  */

#include <stdio.h>
#include <stdlib.h>

extern struct IntuitionBase *IntuitionBase;

void main(void)
{
    /* Revision number for AmigaDOS version 1.3 */
    int ver = 34;

    IntuitionBase = (struct IntuitionBase *)
        OpenLibrary("intuition.library",ver);

    /*  Make sure library opened OK */
    if (IntuitionBase == NULL)
  exit(EXIT_FAILURE);

    /* Insert your code using IntuitionBase here ... */

    CloseLibrary((struct Library *)IntuitionBase);
}
```

1.57 mathbase

MathBase-FFP library vector

Synopsis

```
extern struct Library *MathBase;

MathBase = OpenLibrary("mathffp.library",«revision»);
```

Description

This external location is used to interface with the Motorola Fast Floating Point library routines provided by Amiga. It is initialized by an OpenLibrary call in the __fpinit routine when a program compiled with the math=ffp option starts. It contains the base address of the FFP math library vector table. If you make direct calls to the Amiga FFP functions you must first initialize this location by calling the OpenLibrary function.

You must close the library before your program terminates.

For information about the «revision» parameter, see "Defining Library Bases" in Chapter 2, "Using the Commodore Libraries."

Portability

AmigaDOS

Example

```
/* This example demonstrates how to use OpenLibrary on  */
/* mathffp.library to initialize MathBase.  It finishes */
/* by closing mathffp.library, as all programs which  */
/* open it must do before terminating.      */

#include <stdio.h>
#include <stdlib.h>

extern struct Library *MathBase;

void main(void)
{
    /* Revision number for AmigaDOS version 1.3 */
    int ver = 34;

    MathBase = OpenLibrary("mathffp.library",ver);

    /*  Make sure library opened OK */
    if (MathBase == NULL)
  exit(EXIT_FAILURE);

    /*  Insert your code using Amiga FFP functions here ...  */

    CloseLibrary(MathBase);
}
```

1.58 mathtransbase

MathTransBase-FFP transcendental library vector

Synopsis

```
extern struct Library *MathTransBase;

MathTransBase = OpenLibrary("mathtrans.library",«revision»);
```

Description

This external location is used to interface with the Motorola Fast Floating Point format transcendental function library routines provided by Amiga. It is initialized by an OpenLibrary call when a program compiled with the FFP option starts. It contains the base address of the FFP transcendental math library vector table. If you make direct calls to the Amiga FFP transcendental functions you must first initialize this location by calling the OpenLibrary function.

You must close the library before your program terminates.

For information about the «revision» parameter, see "Defining Library Bases" in Chapter 2, "Using the Commodore Libraries."

Portability

   AmigaDOS

Example

```
  /* This example demonstrates how to use OpenLibrary on */
  /* mathtrans.library to initialize MathTransBase.  It  */
  /* finishes by closing mathtrans.library, as all       */
  /* programs which open it must do before terminating.  */

  #include <stdio.h>
  #include <stdlib.h>

  extern struct Library *MathTransBase;

  void main(void)
  {
      /* Revision number for AmigaDOS version 1.3 */
      int ver = 34;

      MathTransBase = OpenLibrary("mathtrans.library",ver);

      /*  Make sure library opened OK */
      if (MathTransBase == NULL)
    exit(EXIT_FAILURE);

      /*  Insert your code using Motorola Fast Floating Point format
    transcendental functions here ...  */

      CloseLibrary(MathTransBase);
  }
```

1.59 _memtype

_MemType-Type of memory desired

Synopsis

```
  #include <exec/memory.h>

  extern unsigned long _MemType;
```

Description

  The external long integer _MemType represents the type memory to
  be allocated by any of the memory allocation routines. The
  default is MEMF_ANY. You can set _MemType to any of the following
  mnemonics:

```
      Mnemonic      Value
      --------      -----
      MEMF_ANY      0L    (any type of memory will do)
      MEMF_PUBLIC     1L<<0
      MEMF_CHIP     1L<<1
      MEMF_FAST     1L<<2
      MEMF_LOCAL      1L<<8
      MEMF_24BITDMA   1L<<9 (DMAable memory within 24 bits of address)

  These mnemonics are defined in the file exec/memory.h.

  NOTE: It is safe to change _MemType at any point in a program.
```

Portability

   AmigaDOS

Example

```
  #include <exec/memory.h>

  extern long _MemType;

  void main(void)
  {
      void *chipmem;
      void *fastmem;
      void *anymem;
      long oldtype;

      oldtype = _MemType;

      /* allocate 50 bytes of any type of memory */
      _MemType = MEMF_ANY;
      anymem = malloc(50);

      /* allocate 50 bytes of fast memory */
      _MemType = MEMF_FAST;
      fastmem = malloc(50);

      /* allocate 50 bytes of chip memory */
      _MemType = MEMF_CHIP;
      chipmem = malloc(50);

      _MemType = oldtype;
  }
```

See Also

   malloc , AllocMem in the autodocs from Commodore

1.60 __montbl

__montbl-Array of months by name

Synopsis

```
  extern char *__montbl[];
```

Description

  The external array __montbl contains twelve pointers to character strings representing the months of the year. Each string is three bytes long, plus a null terminator. The values are as follows:

Index	String	Meaning
0	Jan	January
1	Feb	February
2	Mar	March
3	Apr	April
4	May	May
5	Jun	June
6	Jul	July
7	Aug	August
8	Sep	September
9	Oct	October
10	Nov	November
11	Dec	December

Portability

SAS/C

Example

```
/*
 *  Replace the month table with a German version
 *  You might do this as a static DATA intialization if you
 *  know you will be in German;  or, you could change this into
 *  a function that would be called if and when the user
 *  selects German from a menu item somewhere
 */

extern char *__montbl[];

void main(void)
{
    __montbl[0] = "Jan";  /* Januar    (January)   */
    __montbl[1] = "Feb";  /* Februar   (February)  */
    __montbl[2] = "Mar";  /* Marz      (March)     */
    __montbl[3] = "Apr";  /* April     (April)     */
    __montbl[4] = "Mai";  /* Mai       (May)       */
    __montbl[5] = "Jun";  /* Juni      (June)      */
    __montbl[6] = "Jul";  /* Juli      (July)      */
    __montbl[7] = "Aug";  /* August    (August)    */
    __montbl[8] = "Sep";  /* September (September) */
    __montbl[9] = "Okt";  /* Oktober   (October)   */
    __montbl[10] = "Nov";  /* November  (November)  */
    __montbl[11] = "Dez";  /* Dezember  (December)  */
}
```

See Also

__daytbl , __months

1.61 __months

__months-Array of month lengths

Synopsis

```
  extern char __months[];
```

Description

The external array __months contains twelve characters, each representing the number of days in a specific month (in a non-leap year.) The values are as follows:

Index	Value	Month
0	31	January
1	28	February
2	31	March
3	30	April
4	31	May
5	30	June
6	31	July
7	31	August
8	30	September
9	31	October
10	30	November
11	31	December

Your programs must account for leap years when dealing with February.

Portability

SAS/C

Example

```
  /*
   *  This is a sample program demonstrating the use of __months[].
   *  Count the number of days since a given month and day.  Assume
   *  both dates are within the same calendar year.  Assume month and
   *  date are BEFORE thismonth and thisdate.
   */

  #include <stdio.h>

  extern char __months[];

  int countdays(int month, int date, int thismonth, int thisdate)
  {
      int days;
      int curmonth;

      if (thismonth == month)
    days = thisdate - date;
```

```
      else
      {
    days = __months[month] - date;

    for (curmonth = month + 1; thismonth > curmonth; curmonth++)
    {
        days += __months[curmonth];
    }

    days += thisdate;
      }
      return(days);
  }

  void main(void)
  {
      int x;

      /* Call countdays, print the result */
      x = countdays(7,12,10,5);
      printf("x = %d \n",x);
  }
```

See Also

   __daytbl , __montbl

1.62 __msflag

__msflag-MS-DOS file pattern flag

Synopsis

```
  extern int __msflag;
```

Description

This external integer is used by the filename pattern matching functions to specify AmigaDOS or MS-DOS wildcard characters. If __msflag is nonzero, MS-DOS filename patterns are used. By default, AmigaDOS filename patterns are used.

Portability

   SAS/C

See Also

   dfind , getfnl

1.63 _mstep

```
_MSTEP-Memory pool increment size

Synopsis

  extern long _MSTEP;

Description

  This external integer is used by the memory allocation functions.
  It specifies the minimum amount of memory that will be allocated
  from the system for the local memory pool.  The default value is
  16,384 bytes.

  While additional memory is added to the local pool, it will not be
  contiguous with the memory already in the pool.  If the additional
  amount is small, it can lead to severe fragmentation of the local
  pool.  The memory allocation functions attempt to avoid this by
  rounding up the amount needed to the next multiple of the figure
  in _MSTEP.

  This technique works well for small allocation requests.  However,
  if your application requires mostly large blocks of memory, _MSTEP
  should be set to a small nonzero figure to allow for a more
  efficient allocation.

Portability

   SAS/C

See Also

   free ,  malloc
```

1.64 __nufbs

```
__nufbs-Count of level 1 file handle slots

Synopsis

  #include <ios1.h>

  extern int __nufbs;

Description

  The external integer __nufbs indicates how many level 1 file
  handles exist in the  __ufbs  linked list.

  In Version 5.10 and earlier, this data name was called _nufbs.

Portability

   SAS/C
```

See Also

__ufbs

1.65 _oserr

_OSERR-DOS error information

Synopsis

```
#include <dos.h>

extern int _OSERR;
```

Description

The external integer _OSERR contains error information returned by AmigaDOS after a system call has failed. In general, the SAS/C library resets _OSERR at the beginning of any function that makes AmigaDOS system calls. Then, if a system call fails during that function, the system error code is saved in _OSERR.

The AmigaDOS error number is mapped into an equivalent UNIX error number, which is placed in errno . If there is no appropriate UNIX number, errno will contain -1, defined symbolically as EOSERR.

The __os_nerr and __os_errlist items are defined in a C source file named oserr.c and are used by the poserr function to print messages that correspond to the code found in _OSERR.

The following list of system error codes applies to AmigaDOS 2.0 and is contained in the file _oserr.c. The #define values for these codes are in the file dos/dos.h.

Code	Meaning
103	not enough memory is available
105	process table is full
114	bad template
115	bad number
116	required argument is missing
117	value after keyword is missing
118	wrong number of arguments
119	unmatched quotes
120	argument line is invalid or too long
121	file is not executable
122	invalid resident library
202	object is in use
203	object already exists
204	directory not found
205	object not found
206	invalid window description
207	object is too large
209	packet request type is unknown
210	object name is invalid

```
     211      invalid object lock
     212      object is not of required type
     213      disk is not validated
     214      disk is write-protected
     215      rename across devices attempted
     216      directory is not empty
     217      too many levels
     218      device (or volume) is not mounted
     219      seek failure
     220      comment is too long
     221      disk is full
     222      object is protected from deletion
     223      file is write-protected
     224      file is read-protected
     225      not a valid DOS disk
     226      no disk is in drive
     232      no more entries are in directory

  The following error codes are defined only under AmigaDOS 2.0 and
  are documented in the dos/dos.h file.

     Code     Meaning
     ----     -------
     233      object is soft link
     234      object is linked
     235      bad loadfile hunk
     236      function not implemented
     240      record is not locked
     241      record lock collision
     242      record lock timeout
     243      record unlock error

  The following errors occur only when calling the MatchFirst
  MatchNext functions of dos.library under AmigaDOS 2.0.  These
  errors are documented in the file dos/dosasl.h.

     Code     Meaning
     ----     -------
     303      buffer overflow
     304      break character received
     305      not executable
```

Portability

   AmigaDOS

See Also

  AmigaDOS Technical Reference Manual, errno , poserr

1.66 __os_errlist

__os_errlist-Array of operating system error messages

Synopsis

```
  struct DOS_ERRS __os_errlist[];
```

Description

  See the _OSERR description for details.

Portability

   SAS/C

See Also

   _OSERR , __os_nerr , poserr

1.67 __oslibversion

__oslibversion-Kickstart version number

Synopsis

```
  #include <dos.h>

  long __oslibversion;
```

Description

If you declare but do not define a system library base in your own code, the library base is automatically initialized. The autoinitialization code calls OpenLibrary to initialize the library base. The autoinitialization functions pass the value of __oslibversion to OpenLibrary. If your program requires a specific version of the operating system, declare this variable in your code and initialize it as necessary. For example, if your program requires Version 2.0 of the AmigaDOS operating system, which is library revision 37, you can enter the following line in your program external to any function:

```
      long __oslibversion = 37;
```

If your program is run under older versions of AmigaDOS, the autoinitialized libraries cannot be opened, and the library function __autoopenfail is called. For more information, refer to Chapter 10, "Using Startup Modules, Autoinitialization, and Autotermination Functions," in SAS/C Development System User's Guide, Volume 1.

The following table lists the library revision numbers for each version of the operating system.

```
      OS        Library
      Version   Revision
      -------   --------
        1.2  33
        1.3  34
        2.0  36  (Preliminary 2.0)
        2.04   37
        2.1  38
        3.0  39
        3.1  40
```

Portability

```
   SAS/C
```

Example

```
  extern long __oslibversion = 37;

  void main(void)
  {
      /* Insert your program here */
  }
```

1.68 __os_nerr

__os_nerr-Highest valid error number in the __os_errlist array

Synopsis

```
  int __os_nerr;       /* Highest valid error number */
```

Description

See the _OSERR description for details.

Portability

SAS/C

See Also

_OSERR , __os_errlist , poserr

1.69 __priority

__priority-Default priority for a background program

Synopsis

```
  #include <dos.h>

  long __priority=-5;
```

Description

The global variable __priority is used by the cback.o startup code

```
  to indicate the priority at which to start the background program.
  A value of 0 is the normal priority while larger values run at
  higher priority.  This number is limited to the range -128 to 127,
  inclusive.

  You must initialize the variable at compile time.  Setting this
  variable after the program has started does not affect the program
  priority.

  If you do not declare a __priority variable, cback.o defaults to a
  priority of 0 (based on a default value drawn from the library).

  If you do not link with cback.o, this variable is ignored.
```

Portability

```
   AmigaDOS
```

Example

```
  /* A priority of 50 will ensure that your background program
   * runs.  A priority this high will even prevent input
   * handlers from running, so the machine will be effectively
   * crashed by doing this (until your program calls
   * something that causes it to pause).
   *
   **** REMEMBER:  You must link with cback.o for this variable
   ****    to have any effect.
   */

  long __priority=50;

  void main(void)
  {
      /* Insert your program here */
  }
```

See Also

```
   __procname ,  __stack
```

1.70 __procname

__procname-Process name for a spawned program

Synopsis

```
  #include <dos.h>

  char *__procname="MyBackgroundProcess";
```

Description

```
  The global variable __procname is the name that cback.o uses when
  spawning a background process.  You may choose any name but should
```

stick to something that is meaningful for your program.

You must initialize this variable at compile time. Setting this variable after the program has started has no effect on the program name. Also, if the program is started from the workbench, the __procname parameter is ignored.

If you do not link with cback.o, this variable is ignored.

Portability

AmigaDOS

Example

```
/*
 * Set the name of the background process.
 *
 ***** REMEMBER:  You must link with cback.o for this variable
 *****      to have any effect.
 */

char *__procname="SpecialProcess";

void main(void)
{
    /* Insert your program here */
}
```

See Also

__priority , __stack

1.71 __sigfunc()

__sigfunc-Signal function table

Synopsis

```
#include <signal.h>

extern void (*__sigfunc[NSIG])(int);
```

Description

__sigfunc is the array of all signal handles. When the raise function is called, it looks in the table to see what is installed at the corresponding slot in the __sigfunc array. Each entry is either a pointer to the function to be called when the event occurs or one of two special values:

```
    SIG_IGN - ignore the condition.
    SIG_DFL - take the default action.
```

There are NSIG elements in the array corresponding to the predefined signal values from the file signal.h.

Symbol	Value	Meaning
SIGABRT	1	abnormal termination, abort
SIGFPE	2	floating-point exception
SIGILL	3	illegal instruction
SIGINT	4	interrupt from AmigaDOS, Control-C or Control-D
SIGSEGV	5	segmentation violation (not generated on the Amiga)
SIGTERM	6	termination request

Signals 0, 7, and 8 are reserved for future use (and for compatibility with POSIX implementations).

You should not set elements of this array directly. Use the signal function instead.

Portability

UNIX

See Also

raise , signal

1.72 _slash

_SLASH-Directory separator character

Synopsis

```
extern char _SLASH;
```

Description

This external character is used by various functions that construct file names. It specifies the character to be used for separating components of the directory path. For AmigaDOS, the default character is a forward slash (/), while for MSDOS it is a backslash (\).

Portability

SAS/C

See Also

strmfn() , strmfp()

1.73 __stack

```
__stack-Minimum program stack size

Synopsis

  #include <dos.h>

  long __stack = 8192;

Description

  The global variable __stack is used by the startup code to
  determine the minimum stack space necessary to run a program.  You
  must initialize the variable at compile time.  Setting this
  variable after the program has started does not affect the stack
  size.

  If the startup code determines that there is not enough stack
  space to satisfy the request, it allocates a temporary stack and
  runs the program on that stack.

Portability

   SAS/C

Example

  /* Make sure we have a LOT of stack space to run. */

  #include <dos.h>

  long __stack = 25000L;

  void main(void)
  {
      /* Insert your program here */
  }

See Also

   __priority ,  __procname
```

1.74 __stdiov37

```
__stdiov37-Standard I/O window behavior array

Synopsis

  #include <dos.h>

  char __stdiov37[] = "/AUTO/CLOSE/WAIT";

Description

  When you run a program from the Workbench, the compiler opens a
```

standard I/O window in which your program can read from and write to stdin, stdout, and stderr. The AmigaDOS Open function is called to open this window. Open uses the character string in __stdiowin to initialize the I/O window. If you are running under Kickstart version 37 or higher (AmigaDOS 2.0 or higher), the contents of __stdiov37 is appended to __stdiowin . The __stdiov37 variable allows you to control the behavior of the standard I/O window.

The AmigaDOS 2.0 console device supports the following keywords that control the behavior of the window:

AUTO
specifies not to open the window unless I/O occurs.

CLOSE
places a Close gadget on the window.

WAIT
specifies not to close the window until you click on the Close gadget or type Control-\.

WINDOW 0x«address»
specifies to use the window pointed to by the «address». Specify the address in hexadecimal notation.

SCREEN name
opens the standard I/O window on the public screen specified.

You can also specify BACKDROP, NODRAG, NOBORDER, NOSIZE, SIMPLE, and SMART. These keywords control the same attributes as the similarly-named Intuition window flags.

The default string specifies that the window should open only if your program actually reads or writes data to it; that the window should have a Close gadget; and that the window should wait for the user to press the Close gadget or type an end-of-file character before closing the window.

For more information on console specifications, refer to the information about AmigaDOS CON: device input and output in The AmigaDOS Manual, 3rd Edition (Commodore-Amiga, Inc. 1991).
To change the window behavior, include a line similar to the line shown in the Synopsis above in any C source file in your project. Declare this variable external to any function, and be sure to statically initialize the variable. Any changes made to this external variable after your program starts do not affect the window.

For more information on managing the standard I/O window, refer to Chapter 9, "Running Your Program from the Workbench," in SAS/C Development System User's Guide, Volume 1.

Portability

SAS/C

Example

```
  /* Define a window that opens only if your program  */
  /* reads or writes data to it and that closes     */
  /* automatically when the program ends.       */

  #include <dos.h>

  char __stdiov37[] = "/AUTO";

  void main(void)
  {
      /* Insert your program here */
  }
```

See Also

   __stdiowin

1.75 __stdiowin

__stdiowin-Standard I/O window attributes array

Synopsis

```
  #include <dos.h>

  char __stdiowin[] = "CON:10/10/320/80/";
```

Description

When you run a program from the Workbench, the compiler opens a standard I/O window in which your program can read from and write to stdin, stdout, and stderr. The AmigaDOS Open function is called to open this window. Open uses the character string in __stdiowin to initialize the I/O window.

The default specification opens a console window starting at location (10,10) that is 320 pixels wide and 80 pixels high.

For more information on console specifications, refer to The AmigaDOS Manual, 3rd Edition (Commodore-Amiga, Inc. 1991).

You can control the attributes of this window using the __stdiowin variable. To change the window attributes, include a line similar to the line shown in the Synopsis above in any C source file in your project. Declare this variable external to any function, and be sure to statically initialize the variable. Any changes made to this external variable after your program starts do not affect the window.

NOTE: The replacement string for __stdiowin must end in a slash (/), or it must end with a window title. If it ends in a slash, the name of your program will be used as the window title. Refer to the examples later in this section for specific examples.

For more information on managing the standard I/O window, refer to Chapter 9, "Running Your Program from the Workbench," in SAS/C Development System User’s Guide, Volume 1.

Portability

SAS/C

Examples

```
/* Make the window 600 pixels wide and 100 pixels */
/* tall.  Add a window title of "My Window."      */

char __stdiowin[]= "CON:10/10/600/100/My Window";


/* Make the window 500 pixels wide and 200 pixels */
/* tall.  Window title will be the name of the    */
/* program.           */

char __stdiowin[]="CON:10/10/500/200/";
```

See Also

__stdiov37

1.76 __stkneed

__STKNEED-Minimum function stack size

Synopsis

```
#include <dos.h>

long __STKNEED = 400;
```

Description

The variable __STKNEED specifies the minimum amount of stack needed by each function in your program.

If you declare a function with the __stackext keyword or compile it with the stackext option, the compiler generates extra code at the start of the function. This extra code compares the amount of stack available with the amount specified in __STKNEED and, if enough stack is not available, allocates a new stack extent whose size is specified in __stack .

The default value of __STKNEED is 400.

The default value of __stack is 8192 bytes.

If a system interrupt occurs, it will use your stack; therefore,

```
  you must keep at least 400 bytes of stack free at all times.

  If your functions require additional space, declare __STKNEED in
  your code and statically initialize it to the amount you require.
  If necessary, you can change the value of __STKNEED in your
  program code; if the current stack does not meet the new __STKNEED
  value, a new one is allocated the next time you call a function
  declared with the __stackext keyword or compiled with the stackext
  option.

  For more information on managing stack space, refer to Chapter 11,
  "Using SAS/C Extensions to the C Language," in SAS/C Development
  System User's Guide, Volume 1.

Portability

   SAS/C

Example

  /* Make sure each function has enough stack space. */

  #include <dos.h>

  long __STKNEED = 600;

  void main(void)
  {
      /* Insert your program here */
  }

See Also

   __stack
```

1.77 _strict_ansi

```
_STRICT_ANSI-Define to disable non-ANSI features

Synopsis

  #define _STRICT_ANSI

Portibility

  SAS/C

Description

  To be completely ANSI-compliant, an implementation must not define
  any non-ANSI symbols in an ANSI header file unless the symbols
  begin with an underscore followed by a capital letter, or an
  underscore followed by another underscore.

  Defining this symbol will prevent the definition of any symbols
  from the SAS/C Development System headers which do not comply with
  the ANSI standard.
```

1.78 sysbase

```
SysBase-Base of the Exec library

Synopsis

  #define __USE_SYSBASE 1

  #include <proto/exec.h>

Description

  This external location is used to point to the base of the Amiga
  operating system library, Exec.  Memory location 4 always contains
  a pointer to the Exec library's structure, and it can be used
  instead of SysBase.  However, on machines with 68020 or higher
  processors, it is faster to use SysBase rather than read location
  4 repeatedly.

  If you use SysBase, the standard startup code for programs and
  libraries initialize SysBase for you.  If you do not link with one
  of the startup modules supplied by SAS/C, you must initialize
  SysBase in your code.

  SysBase is declared in the file proto/exec.h.  To use SysBase
  rather than location 4 in your Exec calls, you should define the
  preprocessor symbol __USE_SYSBASE before including proto/exec.h.

  You cannot obtain the address of the Exec library by using
  OpenLibrary, since OpenLibrary is itself an Exec function.
```

1.79 __sys_errlist

```
__sys_errlist-Errno text strings

Synopsis

  #include <errno.h>

  extern char *__sys_errlist[];

Description

  The external array of strings __sys_errlist provides the text
  message that corresponds to a given  errno  value.  The value of
   errno  is used as a direct index to this table.  You should avoid
  accessing this table directly to print errors.  Instead, use the
   perror  or  strerror  functions.
  You may want to substitute your own table for internationalization
  of your program.  The external integer  __sys_nerr  indicates the
  number of entries in __sys_errlist.

  If you change __sys_errlist dynamically in your program, as shown
  in the example below, the changes are only in effect for that
  program while it is running.  To change the messages permanently
  for all programs, modify syserr.c in the source directory, compile
  it, and use the OML to replace it in all nonmath libraries that
  you plan to use.  As an alternative, you can compile the syserr.c
  file and link it with all programs in which you want modified
  messages.

  In Version 5.10 and earlier, this variable was named sys_errlist.

Portability

   SAS/C

Example

  /* Print error message number 1 with perror, change the */
  /* message to all capitals, and then print the    */
  /* message again.          */

  #include <errno.h>

  void main(void)
  {
      const char *x = "Test Message";

      errno = 1;
      perror(x);
      __sys_errlist[errno] = "USER IS NOT OWNER";
      perror(x);
  }

See Also

   perror ,  strerror ,  __sys_nerr
```

1.80 __sys_nerr

```
__sys_nerr-Number of entries in __sys_errlist

Synopsis

  #include <errno.h>

  extern int __sys_nerr;

Description

  The external integer __sys_nerr indicates the number of entries in
  the  __sys_errlist  array.
  In Version 5.10 and earlier, this variable was named sys_nerr.
```

Portability

```
   SAS/C
```

Example

```
  /*
   * Print all possible errors from the __sys_errlist array
   */

  #include <stdio.h>
  #include <errno.h>

  void main(void)
  {
      int i;

      for ( i=0; i <= __sys_nerr; i++)
      {
    printf("Message %d = %s \n",i,strerror(i));
      }
  }
```

See Also

```
   perror ,  strerror ,  __sys_errlist
```

1.81 __timezone

__timezone-timezone bias from GMT

Synopsis

```
  extern long __timezone;
```

Description

```
  __timezone contains the number of seconds that must be subtracted
  from GMT.  This variable is initialized by the  __tzset  function
  and used by the  localtime  function to adjust from Greenwich Mean
  Time (GMT) to the local time.
```

Portability

```
   UNIX
```

See Also

```
   __daylight ,  localtime ,  __tzname ,
   __tzdtn ,  __tzset ,  __tzstn
```

1.82 _tz

```
_TZ-Default time zone name

Synopsis

  #include <time.h>

  extern char *_TZ;

Description

  This variable is the default time zone that is used if there is no
  environment variable TZ set on the Amiga.  If you set it, you must
  make sure that the memory for it is not freed until you reset it.
  If _TZ is not set to any value, CST6 is used instead.

  The value of the variable _TZ is used to set other time zone
  variables, including  __tzstn ,  __tzdtn , and  __daylight .   The
  first three characters are taken as the name.  The remaining
  characters are taken as the ASCII representation of the number of
  hours offset from Greenwich Mean Time (GMT) for the time zone.

Portability

   SAS/C

Example

  /*
   * Print out the default time zone and then set it to be EST.
   */

  #include <stdio.h>
  #include <time.h>

  void main(void)
  {
      printf("Default Time Zone = %s\n", _TZ);
      _TZ = "EST5";
  }

See Also

   __tzset
```

1.83 __tzdtn

```
__tzdtn-Daylight time name

Synopsis

  extern char __tzdtn[4];
```

Description

```
  __tzdtn contains the three-character name for daylight time.  This
  variable is initialized by the  __tzset  function and used by the
   localtime  function to adjust from Greenwich Mean Time (GMT) to the
  local time.
```

Portability

```
   SAS/C
```

See Also

```
   __daylight ,  localtime ,  __timezone ,
   __tzname ,  __tzset ,  __tzstn
```

1.84 __tzname

__tzname-timezone names

Synopsis

```
  extern char *__tzname[2];
```

Description

```
  The two __tzname pointers point to  __tzstn  and  __tzdtn ,
  respectively. These strings contain the three-character names for
  standard time ( __tzstn ) and daylight time ( __tzdtn ).  These
  variables are initialized by the  __tzset  function and used by
  the  localtime  function to adjust from Greenwich Mean Time (GMT)
  to the local time.
```

Portability

```
   UNIX
```

See Also

```
   __daylight ,  localtime ,  __timezone ,
   __tzdtn ,  __tzset ,  __tzstn
```

1.85 __tzstn

__tzstn-Standard time name

Synopsis

```
  extern char __tzstn[4];
```

Description

__tzstn contains the three-character name for standard time. This variable is initialized by the __tzset function and used by the localtime function to adjust from Greenwich Mean Time (GMT) to local time.

Portability

SAS/C

See Also

__daylight , localtime , __timezone ,
__tzname , __tzdtn , __tzset

1.86 __ufbs

__ufbs-Array of open level 1 file handles

Synopsis

```
#include <ios1.h>

extern struct UFB *__ufbs;
```

Description

__ufbs is used to track all open level 1 file handles. __ufbs is the pointer to the beginning of the linked list containing the file handles. If __ufbs is equal to NULL, then there are no open level 1 files.

The open or creat function returns the entry number of the file handle in the linked list. For example, if the open function returns 5, the file handle is the fifth entry of the linked list.

It is not recommended that you access this linked list directly. The chkufb function is provided to translate an index into the appropriate UFB structure .

In Version 5.10 and earlier, the linked list __ufbs was an array named _ufbs.

Portability

SAS/C

See Also

chkufb , __nufbs , open

1.87 _wbenchmsg

_WBenchMsg-Workbench startup message

Synopsis

```
  #include <dos.h>

  struct WBStartup *_WBenchMsg;
```

Description

_WBenchMsg contains the arguments passed to your program from the Workbench. The value is passed to your program as argv[0] if it is invoked from the Workbench.

For more information about handling arguments passed to your program from the Workbench, refer to Chapter 9, "Running Your Program from the Workbench," in SAS/C Development System User's Guide, Volume 1.

In Version 5.10, this variable was named WBenchMsg.

Portability

AmigaDOS

See Also

main

1.88 abort()

abort-Abort the current process

Synopsis

```
  #include <stdlib.h>

  void abort(void);
```

Description

This function aborts the current process and returns a completion code of 3 to the parent process. Also, the message Abnormal program termination is sent to stderr. Level 2 I/O buffers are flushed.

From within a shared library, you must not call any library functions that terminate your program. For example, you cannot call exit , __exit , or abort from a shared library. You also cannot use setjmp and longjmp to jump across a call from the program into the library.

Portability

ANSI

Example

```
  /* Do your own memory allocation */

  #include <stdio.h>
  #include <stdlib.h>

  void *gmem(size_t size)
  {
      void *p;

      if ((p=malloc(size))==NULL)
      {
    abort();
      }
      return(p);
  }

  void main(void)
  {
      void *p;

      p = gmem(4096);
      if (p)
      {
    free(p);
      }
  }
```

See Also

exit , __exit , raise , _XCEXIT

1.89 abs()

abs-Absolute value

Synopsis

```
  #include <stdlib.h>

  ax = abs(x);

  type x;
  type ax;
```

Description

This macro computes the absolute value of the various numeric data types. It accepts any data type as its argument and generates in-line code to perform the conversion. The definition is

```
    #define abs(x) ((x)<0?-(x):(x))
```

  To minimize code size, you can use the fabs , iabs , or labs
  function instead. Choose the function that corresponds to the
  data type being converted.

Portability

   ANSI

Returns

  The return value is the absolute value of the argument.

See Also

   fabs , iabs , labs

1.90 access()

access-Check file accessibility

Synopsis

```
  #include <stdio.h>

  ret = access(name,mode);

  int ret;      /* return code */
  const char *name;  /* file name */
  int mode;     /* access mode */
```

Description

  This function checks if a file is accessible in the way specified
  by the mode parameter. The following table shows the modes, which
  follow the UNIX format, accepted by this function.

Name	Value	Meaning
R_OK	4	check if the file is readable
W_OK	2	check if the file is writable
X_OK	1	check if the file is executable
F_OK	0	check if the file exists

  You can simultaneously check for more than one attribute by
  combining these flags using the logical OR operator.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   UNIX

Returns

```
  A return value of 0 indicates that access is allowed.  If access
  is denied or the file cannot be found, -1 is returned.  Additional
  error information can then be found in the external integers  errno
  and  _OSERR .
```

```
Example
```

```
  /* Check to see if the file s:myconfig */
  /* exists and is writable.         */

  #include <stdio.h>

  void main(void)
  {
      if (access("s:myconfig",F_OK|W_OK)==0)
      {
    printf("yes\n");
      }
      else
      {
    printf("no\n");
    exit(EXIT_FAILURE);
      }
  }
```

```
See Also

   chmod ,  errno ,  _OSERR
```

1.91 acos()

```
acos-Arccosine function

Synopsis

  #include <math.h>

  r = acos(x);

  double r;  /* result */
  double x;  /* angle */

Description

  The acos routine computes the arccosine of x and returns the
  angular value expressed in radians.  The result is constrained
  from 0 to pi.  The argument x must be between -1.0 and 1.0,
  inclusive, or a DOMAIN error will be raised.

Portability

   ANSI

Returns
```

This function returns the arccosine of the angle expressed in radians.

See Also

cos , __matherr

1.92 argopt()

argopt-Get options from an argument list

Synopsis

```
  #include <stdlib.h>

  optd = argopt(argc,argv,opts,argn,optc);

  char *optd;      /* option data pointer        */
  int argc;      /* argument count           */
  const char *argv[];  /* argument vector          */
  const char *opts;    /* options expecting data         */
  int *argn;       /* next argument number (changed) */
  char *optc;      /* option character (changed)     */
```

Description

This function examines an argument list to find the next option argument, using the conventions similar to those of the UNIX shell command processor. These conventions follow:

- An option is an argument that begins with a slash (/) or a dash (-) and appears between the command verb (argv[0]) and the first non-option argument. This function recognizes either a slash or a dash because these characters are used by other systems.

- The character immediately following the dash is called the option character, and it may be followed by a character string known as the option data.

- If the option character appears in the opts string, then the data can be separated from the character by white space. In effect, this means that the data might be in the next argv entry if it does not follow the option character in the current entry.

- A dash or slash followed by a blank or a dash indicates the end of the options.

Each time argopt is called, it finds the next option in the argument array and updates the integer referenced by the argn parameter. On the first call, you should set this integer to 1, since argv[0] points to the command verb. The argc and argv items are normally the same as those passed to your main program, and

```
  they are not changed as a result of the argopt calls.  The option
  character is returned in the byte referenced by the optc
  parameter, and the function returns a pointer to the option data
  string or to a NULL byte.  If the next entry in the argv vector is
  not an option, then the function returns a NULL pointer.

  The opts item provides some flexibility in the way the option data
  are handled.  If the opts parameter points to an empty string,
  then any option data must immediately follow the option character.
  However, if the opts parameter is not empty, then it lists the
  option characters that always have data.  For those characters,
  the data can be preceded by white space in the command line.  What
  this actually means is that the argopt function will look at the
  next entry in the argv vector if the option character is not
  followed by a data string.  If that next entry does not begin with
  a dash, then it is taken as the option data.  See the examples
  below for clarification.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

Returns

  If the next argument is not an option, the function returns a NULL
  pointer.  Otherwise, it returns a pointer to the option data,
  which will be an empty string if there were no data.  Also, if an
  option was found, the character is placed into the byte referenced
  by the optc parameter, and the argn parameter is adjusted to index
  the next entry in the argv vector.

Example

  /*
   * Assume that this program is invoked with the following
   * command line:
   *
   *   myprog -x -ypdq -z -g moo -g - blah
   *
   * The output will then be:
   *
   *    Option: x Data:
   *    Option: y Data: pdq
   *    Option: z Data:
   *    Option: m Data: oo
   *    Option: g Data:
   *    Arg[8]: blah
   */
  #include <stdio.h>
  #include <stdlib.h>

  char opts[] = "gx";

  void main(int argc, char *argv[])
  {
      char option,*odata;
      int next;

      next = 1;
      while((odata = argopt(argc,argv,opts,&next,&option)) != NULL)
      {
    printf("Option: %c, Data: %s\n",option,odata);
      }
      for (; next < argc; next++)
      {
    printf("Arg[%d]: %s\n",next,argv[next]);
      }
  }
```

See Also

   main

1.93 asctime()

asctime-Generate an ASCII time string

Synopsis

```
  #include <time.h>

  s = asctime(t);

  char *s;    /*points to time string    */
  const struct tm *t; /*points to time structure */
```

Description

This function converts a time structure into an ASCII string. The time structure argument t is usually returned by the gmtime or localtime function.

Portability

   ANSI

Returns

This function returns an ASCII string of exactly 26 characters having the form:

```
      "DDD MMM dd hh:mm:ss YYYY\n\0"
```

DDD is the day of the week, MMM is the month, dd is the day of the month, hh:mm:ss is the hour:minute:seconds, and YYYY is the year. An example is

```
      "Wed Sep 04 15:13:22 1985\n\0"
```

```
  The time pointer returned by the function refers to a static data
  area that is shared by both the  ctime  and asctime functions.
```

Example

```
  #include <stdio.h>
  #include <time.h>

  void main(void)
  {
      struct tm *tp;
      long t;

      time(&t);
      tp = localtime(&t);
      printf("Current time is %s\n",asctime(tp));
  }
```

See Also

```
   ctime ,  gmtime ,  localtime ,  strftime
```

1.94 asin()

asin-Arcsine function

Synopsis

```
  #include <math.h>

  r = asin(x);

  double r;    /* result*/
  double x;    /* angle */
```

Description

```
  The asin routine computes the arcsine and returns an angular value
  expressed in radians.  The result is constrained as -pi/2 to pi/2.
  The argument must be between -1.0 and 1.0, inclusive, or a DOMAIN
  error will be raised.
```

Portability

```
   ANSI
```

Returns

```
  This function returns the arcsine of the argument expressed in
  radians.
```

See Also

```
   __matherr ,  sin
```

1.95 assert()

```
assert-Assert program validity

Synopsis

  #include <assert.h>

  assert(x);
  __assert(x,xs,file,line);

  int x;        /* expression to check for nonzero value */
  const char *xs;     /* assertion in text form */
  const char *file;   /* source file name */
  int *line;      /* source line number */

Description

  The assert macro tests an expression x for validity (nonzero
  value).  If a condition in your program is false (0), the assert
  macro is a quick way to abort the program and print an error
  message to stderr.  If the expression is false, then the macro
  calls the __assert function with the expression in text form plus
  the source filename (as defined in the __FILE__ macro) and line
  number (as defined in the __LINE__ macro), also as text strings.
  The default version of the __assert function prints a message to
  stderr and aborts with an exit code of 1.

  NOTE:  You cannot call assert or __assert from a shared library.

  To define the macro, include the  assert.h header file in your
  program.  The assert.h file contains two versions of the macro, a
  null version and the normal code-generating version.  Using the
  null version allows you to strip the assertion code from your
  program without removing the assert calls.  To use the null
  version, define the symbol NDEBUG in one of your header files.  If
  you define NDEBUG in one of your header files, the header file
  containing the NDEBUG definition must be included before the
  assert.h file.  If the symbol NDEBUG is defined, the null version
  of the macro is used. If the symbol NDEBUG is not defined, the
  normal code-generating version applies.

Portability

   ANSI

Example

  #include <stdio.h>
  #include <assert.h>

  /* Make sure integer x is positive */
  void postest(int x)
  {
      assert(x >= 0);
  }

  void main(void)
  {
      postest(5);
      printf("5 is a positive integer\n");
  }
```

1.96 astcsma()

astcsma-AmigaDOS string pattern match (anchored)

Synopsis

```
  #include <string.h>

  length = astcsma(s,p);

  int length;   /* length of matching string */
  const char *s;    /* string being scanned */
  const char *p;    /* pattern string */
```

Description

The function astcsma performs a simple anchored AmigaDOS style pattern match. That is, you can specify the wildcards #? in the pattern string. The pattern must match at the beginning of the specified string.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

AmigaDOS

Returns

This function returns the length of the matching string or 0 if there was no match.

Example

```
  /* Show all files matching the pattern "#?.c" */

  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #include <sys/types.h>
  #include <sys/dir.h>
  void main(void)
  {
      DIR *dfd;
      struct direct *dptr;

      dfd=opendir("");  /* opens the current directory */
      if (dfd != NULL)
      {
    while ((dptr=readdir(dfd))!=NULL)
    {
        if (astcsma(dptr->d_name,"#?.c")!=0)
        {
      printf("%s\n",dptr->d_name);
        }
    }
    closedir(dfd);
      }
  }
```

See Also

   stcpm , stcpma , stcsma

1.97 atan()

atan-Arctangent function

Synopsis

```
  #include <math.h>

  r = atan(x);

  double r;    /* result */
  double x;    /* angle */
```

Description

  The atan routine computes the arctangent of x and returns an
  angular value expressed in radians. The result is constrained as
  -pi to pi.

  Since the tangent becomes very large for angles close to pi/2, the
   atan2 function is often used to avoid computations with large
  numbers that might easily overflow. With the atan2 function, you
  can express the large tangent value as a quotient of two more
  reasonable numbers.

Portability

   ANSI

Returns

  This function returns the arctangent of the argument expressed in radians.

See Also

   atan2 , __matherr , tan

1.98 atan2()

atan2-Arctangent of x/y

Synopsis

```
  #include <math.h>

  r = atan2(x,y);

  double r;   /* result; */
  double x,y; /* angle */
```

Description

  The atan2 routine computes the arctangent of x/y and returns an
  angular value expressed in radians. The result is constrained as
  -pi to pi.

  You can express the large tangent value as a quotient of two more
  reasonable numbers.

  Since the tangent becomes very large for angles close to pi/2, the
  atan2 function is often used to avoid computations with large
  numbers that might easily overflow.

Portability

   ANSI

Returns

  This function returns the arctangent of the argument expressed in
  radians.

See Also

   atan , __matherr , tan

1.99 atexit()

atexit-Set an exit trap

Synopsis

```
  #include <stdlib.h>
  error = atexit(func);

  int error;          /* zero for success  */
  void (*func)(void);    /* trap function pointer */
```

Description

  This function registers an exit trap, which will be called when
  the program exits. There is no limit to the number of exit
  functions you can have. Exit functions are called in last in,
  first out (LIFO) order.

  After you establish an exit routine, you cannot prevent it from
  being called. All functions will be called, in reverse order of
  their registration, when the program exits.

  At the time the exit traps are called, all files opened with the
   open or fopen function are still open, unless specifically closed
  by your code. All memory allocated with the malloc function and
  other ANSI memory-allocation functions is still allocated unless
  specifically freed by your code. After the exit traps have been
  called, these resources are freed.

Portability

   ANSI

Returns

  A zero is returned for success and a nonzero value is returned if
  an error is encountered.

See Also

   exit

1.100 atof()

atof-Convert an ASCII string to a floating-point number

Synopsis

```
  #include <stdlib.h>

  d = atof(p);

  double d;  /* floating point result */
  const char *p;   /* input string pointer */
```

Description

  This function converts an ASCII input string into a
  double-precision floating-point number. The string can contain
  leading white space and a plus or minus sign, followed by a valid

```
  floating-point number in normal or scientific notation.  If
  scientific notation is used, there can be no white space between
  the number and the exponent.  For example, the following is a
  valid number in scientific notation:

      123.456e-53
```

```
Portability

   ANSI

Returns

  This function returns the double-precision floating-point
  equivalent of the ASCII string.

Example

  /* This program tests the atof function. */

  #include <stdio.h>
  #include <math.h>

  void main(void)
  {
      char buff[80];
      double d;

      while(1)
      {
    printf("\nEnter a number: ");
    if (fgets(buff,sizeof(buff),stdin) == NULL)
    {
        break;
    }
    if (buff[0] == '\0')
    {
        break;
    }
    d = atof(buff);
    printf("%e\n",d);
      }
      printf("\n");
  }

See Also

   atoi ,  atol ,  stcd_i ,  stcd_l ,  strtod
```

1.101 atoi()

```
atoi-Convert an ASCII string to an integer

Synopsis
```

```
  #include <stdlib.h>

  x = atoi(s);

  int x;
  const char *s;
```

Description

  This function converts an ASCII string into a normal integer. The
  string must have the form:

      [whitespace][sign]digits

  Whitespace indicates optional leading white space, sign indicates
  an optional + or - sign character, and digits is a contiguous
  string of digit characters. Once the digit portion is reached,
  the conversion continues until a nondigit character is reached.
  No check is made for integer overflow.

Portability

   ANSI

Returns

  This function returns the integer equivalent of the ASCII string.

See Also

   atof , atol , stcd_i , stcd_l , strtod , strtol

1.102 atol()

atol-Convert an ASCII string to a long integer

Synopsis

```
  #include <stdlib.h>

  y = atol(s);

  long int y;
  const char *s;
```

Description

  This function converts ASCII strings into long integers. The
  string must have the form:

      [whitespace][sign]digits

  Whitespace indicates optional leading white space, sign indicates
  an optional + or - sign character, and digits is a contiguous
  string of digit characters. Once the digit portion is reached,

```
  the conversion continues until a nondigit character is reached.
  No check is made for integer overflow.

Portability

   ANSI

Returns

  This function returns the long integer equivalent of the ASCII
  string.

See Also

   atof ,  atoi ,  stcd_i ,  stcd_l ,  strtod ,  strtol
```

1.103 __autoopenfail()

```
__autoopenfail-Terminate program if OpenLibrary fails

Synopsis

  void __autoopenfail(char *lib);

Description

  If you declare but do not define a system library base in your own
  code, the library base is automatically initialized.  If the
  autoinitialized libraries cannot be opened, __autoopenfail is
  called.  By default, __autoopenfail prints a message indicating
  which library could not be opened and then terminates your
  program.

  The source code for __autoopenfail is in sc:source/autoopenfail.c.

  For complete information on initializing system library bases,
  refer to Chapter 10, "Using Startup Modules, Autoinitialization,
  and Autotermination Functions," in SAS/C Development System User’s
  Guide, Volume 1.

Portability

   AmigaDOS

See Also

   __oslibversion
```

1.104 bldmem()

```
bldmem-Build a memory pool of a specified size
```

Synopsis

```
  #include <stdlib.h>

  i=bldmem(n);

  int n;      /* number of 1K-byte blocks in pool */
  int i;      /* -1 for failure */
```

Description

  The bldmem function uses the sbrk function to get up to n
  contiguous 1 kilobyte blocks of memory for the pool. If n is 0,
  the pool is initialized, but no memory is allocated.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   OLD

Returns

  This function returns a -1 if the sbrk function fails.

See Also

   getmem , getml , rlsmem , rlsml , sbrk , sizmem

1.105 bsearch()

bsearch-Perform a binary search

Synopsis

```
  #include <stdlib.h>

  void *bsearch(srch, array, n, size,cmp);

  const void *srch;       /* object searched for    */
  const void *array;      /* pointer to array to search   */
  size_t n;        /* number of members of array   */
  size_t size;         /* size of each element     */
  int (*cmp)(const void *, const void *);
              /* pointer to comparison function */
```

Description

  The bsearch function scans a sorted array pointed to by the array
  pointer for a match with a search value addressed by the srch
  pointer. array is a pointer to the first element of the array to
  be scanned. n is the number of elements in the block, and size is
  the size of each element in bytes.

The bsearch function calls a user-provided comparison function, cmp, passing it pointers to the two objects being compared. The first pointer points to the key. The second pointer points to an array element.

The cmp function returns the following values:

- a negative integer if the first of the two objects is less than the second
- a positive integer if the first object is greater than the second
- 0 if the two objects are equal.

The array to be searched should be sorted in ascending order.

Portability

ANSI

Returns

The bsearch function returns a pointer to the element that matches the search value. If no match can be found, the bsearch function returns a NULL pointer.

See Also

qsort

1.106 calloc()

calloc-Allocate and clear memory

Synopsis

```
#include <stdlib.h>

b = calloc(nelt,esize);

void *b;       /* block pointer      */
size_t nelt;   /* number of elements */
size_t esize; putes the quotient and remainder of the first
argument, y, divided by the second argument, z.
```

Portability

ANSI

Returns

The div function returns a structure of type div_t, which contains both the quotient and remainder. The definition of the div_t type is

```
      typedef struct {
    int rem;
    int quot;
      } div_t;

  The return value is such that

      numer = quot * denom + rem

  The sign of the rem value is the same as the sign of the numer
  value.
```

Example

```
  /* This example converts an angle in radians */
  /* to degrees, minutes, and seconds      */

  #include <stdio.h>
  #include <stdlib.h>
  #include <math.h>

  void main(void)
  {
      double rad, angle;
      int deg, min, sec;
      div_t d;

      rad = 2.414;

      /* Convert angle to seconds and discard fraction */
      angle = rad * (180 * 60 * 60)/PI;

      sec = angle;
      d = div(sec, 60);
      sec = d.rem;

      d = div(d.quot, 60);
      min = d.rem;
      deg = d.quot;

      printf("%f radians = %d degrees, %d minutes, %d seconds\n",
       rad, deg, min, sec);
  }
```

See Also

```
   ldiv
```

1.107 dnext()

dnext-Find the next directory entry

Synopsis

```
  #include <dos.h>

  error = dnext(info);

  int error;        /* 0 if successful */
  struct FileInfoBlock *info; /* file information area */
```

Description

  The dnext function searches a directory for entries that match the specified filename or filename pattern. You must use the dfind function to locate the first matching file. Then, successive calls to the dnext function locate additional matching files. Each dnext call must be given the file information that was returned on the preceding call to the dfind or dnext function.

  The info argument points to a file information structure as defined in the dos.h header file. For AmigaDOS, this is the same as the AmigaDOS FileInfoBlock structure.

  info is a pointer to a file information block that must be allocated on a 4-byte (long word) boundary by the calling program. A common error is failing to allocate the structure before calling the function. You can make sure the structure is long-word aligned by either declaring it with the __aligned keyword or by allocating it dynamically with any SAS/C or system allocation function (such as the malloc or AllocMem functions).

Portability

   AmigaDOS

Returns

  If the operation is successful, a value of 0 is returned. Otherwise, the return value is -1, and you can find additional error information in the external integers errno and _OSERR .

See Also

   dfind , errno , _OSERR

1.108 _dopen()

_dopen-Open an AmigaDOS file

Synopsis

```
  #include <dos.h>

  fh = _dopen(name,mode);

  long fh;      /* file handle (-1 for error) */
  const char *name;   /* file name */
  int mode;     /* access mode */
```

```
Description

  This function opens an AmigaDOS file and returns the file handle.
  The mode argument must be a mode supported directly by AmigaDOS
  and defined in the Amiga header file dos/dos.h.

  Valid modes are as follows:

      MODE_OLDFILE
    opens an existing file with read and write access and
    positions the file pointer at the beginning of the file.

      MODE_NEWFILE
    opens a new file with read and write access and an
    exclusive lock.  Deletes the old file.

      MODE_READWRITE
    opens an old file with a shared lock.  Creates a new file
    if it doesn't exist.

  This function is obsolete and provided for compatibility only.
  Use the AmigaDOS Open function instead.

Portability

   OLD

Returns

  If the operation is successful, the function returns a file
  handle. Otherwise, it returns -1 and places error information in
  the external integers  errno  and  _OSERR .

See Also

   _dclose ,  _dcreat ,  _dcreatx ,  _dread ,  _dseek ,  _dwrite ,
   errno ,  open ,  _OSERR , Open (The AmigaDOS Manual, 3rd Edition)
```

1.109 dos_packet()

```
dos_packet-Sends AmigaDOS output packet

Synopsis

  #include <functions.h>

  retval = dos_packet(struct MsgPort *port, long type, long arg1,
          long arg2, long arg3, long arg4, long arg5,
          long arg6, long arg7);

  long retval;    /* specific to packet type */
  struct MsgPort *port; /* MsgPort to receive the packet */
  long type;    /* type of message to send */
  long argn;    /* arguments for packet */
```

Description

```
  This function sends an AmigaDOS packet to a message port in the
  Amiga system.  The type parameter is the specific packet type you
  want to send, such as CMD_READ, and so on.  The arg1 through arg7
  parameters are argument values specific to the type of packet that
  you are sending.
```

Portability

```
   OLD
```

Returns

```
  The return value is specific to the packet type.
```

See Also

```
  The AmigaDOS Manual, 3rd Edition (AmigaDOS Packets), DoPkt in
  AmigaDOS V2.0 and higher
```

1.110 dqsort()

dqsort-Sort an array of double-precision floating-point numbers

Synopsis

```
  #include <stdlib.h>

  void dqsort(da,n);

  double *da;  /* pointer to beginning of an array of doubles */
  size_t n;    /* number of elements in array */
```

Description

```
  The dqsort function sorts the specified array of double-precision
  floating-point numbers using the ACM 271 algorithm, more popularly
  known as Quicksort.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

```
   SAS/C
```

See Also

```
   fqsort ,  lqsort ,  qsort ,  sqsort ,  tqsort
```

1.111 drand48()

```
drand48-Generate a random double-precision floating-point number (internal
seed)

Synopsis

  #include <math.h>

  x = drand48();

  double x;      /* random double */

Description

  This function generates random numbers using the linear
  congruential algorithm and 48-bit arithmetic.  The drand48
  function uses an internal 48-bit storage area for the seed value.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Returns

  The drand48 function returns double-precision floating-point
  values distributed uniformly over the interval from 0.0 up to but
  not including 1.0.

Portability

   UNIX

See Also

   erand48 ,  jrand48 ,  lcong48 ,  rand ,  srand ,  srand48
```

1.112 _dread()

```
_dread-Read an AmigaDOS file

Synopsis

  #include <dos.h>

  count = _dread(fh,buffer,length);

  unsigned int count;   /* actual number of bytes read */
  long fh;        /* file handle */
  char *buffer;       /* data buffer */
  unsigned int length;  /* number of bytes to read */

Description

  This function reads an AmigaDOS file whose handle was returned by
```

the _dcreat , _dcreatx , or _dopen function. Under normal circumstances, the value returned should match the buffer length. If this value is -1 or greater than the requested length, then some type of error occurred, and you should consult the external integers errno and _OSERR .

If the actual length is less than the requested length when reading, then usually the file is exhausted. It is still a good idea to check the external integers errno and _OSERR in case some malfunction caused the short count.

This function is obsolete and provided for compatibility only. Use the AmigaDOS Read function instead.

Portability

OLD

Returns

If the operation is successful, the function returns the actual number of bytes transferred. Otherwise, it returns -1 and places error information in the external integers errno and _OSERR .

See Also

_dclose , _dcreat , _dcreatx , _dseek , _dwrite , errno , _OSERR , Read (The AmigaDOS Manual, 3rd Edition)

1.113 _dseek()

_dseek-Reposition an AmigaDOS file

Synopsis

```
#include <dos.h>

apos = _dseek(fh,rpos,mode);

long apos;  /*  actual file position */
long fh;    /*  file handle */
long rpos;  /*  relative file position */
int mode;   /*  seek mode */
```

Description

This function repositions an AmigaDOS file whose handle was returned by the _dcreat , _dcreatx , or _dopen function. The seek mode is the same as for the lseek function, as follows:

```
    Mode    Position
    ----    --------
     0      relative to the beginning of the file
     1      relative to the current file location
     2      relative to the end of the file
```

```
  For modes 1 and 2, the rpos argument can be positive or negative,
  but the apos argument is always the actual (positive) position
  relative to the beginning of the file.

  This function is obsolete and provided for compatibility only.
  Use the AmigaDOS Seek function instead.
```

Portability

```
   OLD
```

Returns

```
  If the operation is successful, the function returns the actual
  file position, which is a long integer.  Otherwise, it returns -1L
  and places error information in the external integers  errno  and
   _OSERR .
```

See Also

```
   _dclose ,  _dcreat ,  _dcreatx ,  _dopen ,  _dread ,  _dwrite ,
   errno ,  _OSERR , Seek (The AmigaDOS Manual, 3rd Edition)
```

1.114 _dwrite()

_dwrite-Write to an AmigaDOS file

Synopsis

```
  #include <dos.h>

  count = _dwrite(fh,buffer,length);

  unsigned int count;    /* actual bytes read or written */
  long fh;         /* file handle */
  char *buffer;        /* data buffer*/
  unsigned int length;   /* number of bytes to read or write */
```

Description

```
  This function writes an AmigaDOS file whose handle was returned by
  the  _dcreat ,  _dcreatx  or  _dopen  function. Under normal
  circumstances, the value returned should match the buffer length.
  If this value is -1 or greater than the requested length, then
  some type of error occurred, and you should consult the external
  integers  errno  and  _OSERR .

  If the actual length is less than the requested length when
  writing, then then usually the device has no more space available.
  It is still a good idea to check the external integers  errno  and
   _OSERR  just in case some malfunction caused the short count.

  This function is obsolete and provided for compatibility only.
  Use the AmigaDOS Write function instead.
```

Portability

   OLD

Returns

  If the operation is successful, the function returns the actual
  number of bytes transferred. Otherwise, it returns -1 and places
  error information in the external integers errno and _OSERR .

See Also

   _dclose , _dcreat , _dcreatx , _dread , _dseek , errno ,
   _OSERR , Write (The AmigaDOS Manual, 3rd Edition)

1.115 ecvt()

ecvt-Convert a double-precision floating-point number to a string

Synopsis

```
  #include <math.h>

  s = ecvt(v,dig,decx,sign);

  char *s;   /* string pointer */
  double v;  /* floating point value */
  int dig;   /* number of digits */
  int *decx; /* pointer to decimal index (returned) */
  int *sign; /* pointer to sign indicator */
```

Description

  This function converts a floating-point number into an ASCII
  character string consisting of digits only and terminated by a null
  character.

  The second argument, dig, indicates the total number of digits
  that should be generated. If the floating-point value contains
  fewer significant digits, zeroes are appended. If there are too
  many significant digits, the low-order (right-most) digit is
  rounded.

  The decx argument points to an integer that receives a value
  indicating where the decimal point should be placed in the string.
  For example, an index value of 3 indicates that the decimal point
  should be placed just after the third character in the string. A
  value of 0 means that the decimal point is just before the first
  character. If the index is negative, it indicates the number of
  zeroes that are between the decimal point and the first character.
  For example, -3 means that there are three zeroes between the
  decimal point and the beginning of the string.

  The sign argument points to an integer that is nonzero if the

```
  value v is negative.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

```
Portability

   UNIX

Returns

  This function returns a pointer to a character string containing
  the ASCII equivalent of the float argument.

Example

  #include <stdio.h>
  #include <math.h>

  void main(void)
  {
      int decx, sign;
      char *string;

      string = ecvt(3.1415926535,10,&decx,&sign);

      /* string => "3141592654" */
      /* decx   => 1          */
      /* sign   => 0          */

      printf("string = %s, decx = %d, sign = %d\n",
       string, decx, sign);
  }

See Also

   fcvt
```

1.116 __emit()

```
__emit-Emit 680x0 instruction word

Synopsis

  #include <dos.h>

  void __emit(inst);

  int inst;

Description

  The built-in function emit takes a constant 16-bit integer value
  corresponding to a 68000 assembly language instruction and inserts
  it inline with the code.  However, it does not check whether the
  16-bit value is a valid 68000 instruction.

  The __emit function requires an integer parameter even though the
  assembly language instruction is only 16 bits long.

  CAUTION:  Use this function carefully.  Using this function
  incorrectly can create serious problems.

Portability

   SAS/C

Example

  emit (0x4180)  /* Assembler instruction for chk d0,d0 */

See Also

   getreg ,  putreg
```

1.117 _epilog()

```
_EPILOG-Profiler _EPILOG hook

Synopsis

  #include <sprof.h>

  void __asm _EPILOG(register __a0 const char *where);

Description

  If you compile a function with the profile option,  _PROLOG  and
  _EPILOG are automatically called when the function is entered or
  exited, respectively.   _PROLOG  and _EPILOG were designed for use
  by the sprof utility, but you can replace them with your own code
  and use them for any purpose.

  The SAS/C versions of  _PROLOG  and _EPILOG note the time the
  function was entered and exited and pass this information to the
  sprof utility, which produces a report telling you how much time
  was spent in each function.

  If you declare  _PROLOG  and _EPILOG in your code, make sure you
  declare them with the __asm and register __a0 keywords as shown.
  If you declare either  _PROLOG  or _EPILOG, you must declare both,
  even if one of them simply returns immediately.
  sc:source/profile.c contains the source code for the SAS/C
  versions of  _PROLOG  and _EPILOG and the autoinitialization and
  autotermination functions associated with them.

  The parameter where is passed on the stack.  It points to a
  character string of the following form:

      "\module\function\line"
  where
      module
    is the name of the file containing the function

      function
    is the name of the function

      line
    is the line number on which the function begins.

  For example, if you have a function foo beginning on line 17 of
  the file foo.c, the where parameter would be

      "\foo.c\foo\17"

  A null where parameter indicates that the PROFILER_ON or
  PROFILER_OFF macro has been called. PROFILE_OFF turns off
  profiling for the code that follows it. PROFILE_ON reinstates
  profiling.

  For more information about profiling,  _PROLOG , _EPILOG,
  PROFILE_ON, and PROFILE_OFF, refer to the description of the sprof
  utility in SAS/C Development System User's Guide, Volume 2.

Portability

   SAS/C

See Also

   _PROLOG
```

1.118 erand48()

```
erand48-Generate a random double-precision floating-point number (external
seed)

Synopsis

  #include <math.h>

  x = erand48(seed);

  double x;   /* random double */
  unsigned short seed[3]; /* seed value (high bits in seed[0]) */

Description

  This function generates random numbers using the linear
  congruential algorithm and 48-bit arithmetic.  The erand48
  function is provided for cases where several seeds are in use at
  the same time, so you can specify the seed on each function call.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   UNIX

Returns

  The erand48 function returns double-precision floating-point
  values distributed uniformly over the interval from 0.0 up to but
  not including 1.0.

See Also

   drand48 ,  jrand48 ,  lcong48 ,  rand ,  srand ,  srand48
```

1.119 except()

```
except-Call the math error handler function

Synopsis

  #include <math.h>

  r = except(type,name,arg1,arg2,retval);

  double r;   /* actual return value */
  int type;   /* error type */
  const char *name; /* math function name */
  double arg1;    /* first argument */
  double arg2;    /* second argument */
  double retval;    /* proposed return value */

Description

  The except function is a SAS/C extension to UNIX that simplifies
  the interface to the  __matherr  function by setting up the
  exception vector and processing the action code and return value.
  It is intended to ease the error-handling chore in user-written
  math functions.

  When your math function encounters an error, it should call the
  except function specifying one of the following error types, which
  are defined in the math.h header file:

      Symbol     Code  Meaning
      ------     ----  -------
      _DOMAIN 1    domain error
      _SING 2    singularity
      _OVERFLOW 3    overflow (number too large)
      _UNDERFLOW  4    underflow (number too small)
      _TLOSS  5    total loss of significance
      _RANGE  6    range error
      _PLOSS  7    partial loss of significance
```

If the _STRICT_ANSI flag has been defined, you must use the alternate entry point __except. If the _STRICT_ANSI flag has not been defined, equivalent names without the leading underscore are also defined for compatibility with previous releases of the compiler.

Portability

SAS/C

Returns

The actual return value (a double-precision floating-point value) is passed back.

See Also

__fperr, __matherr

1.120 exit()

exit-Terminate program execution

Synopsis

```
#include <stdlib.h>

void exit(code);

int code;
```

Description

This function terminates the current program and returns to the operating system. Before exiting, any functions specified in a call to the atexit function are called. Next, any open level 1 or level 2 files (opened with the open or fopen function) are closed. Finally, all level 1 and level 2 memory is released back to the system.

Your program must free any memory allocated with AmigaDOS functions and close any file opened with the AmigaDOS functions.

From within a shared library, you must not call any library functions that terminate your program. For example, you cannot call exit, __exit , or abort from a shared library. You also cannot use setjmp and longjmp to jump across a call from the program into the library.

Portability

ANSI

Example

```
/* This example shows how you would abort a program  */
/* if it is not called with a valid input file name. */

#include <stdlib.h>
#include <stdio.h>

void main(int argc, char *argv[])
{
    FILE *f;

    if (argc > 1)
    {
  f = fopen(argv[1],"r");
  if (f == NULL)
  {
      fprintf(stderr, "Can’t open file \"%s\"\n", argv[1]);
      exit(EXIT_FAILURE);
  }
  fclose(f);
    }
    else
    {
  fprintf(stderr,"No file specified\n");
  exit(EXIT_FAILURE);
    }
}
```

See Also

   atexit , longjmp

1.121 __exit()

__exit-Terminate program execution with no clean-up

Synopsis

```
#include <stdlib.h>

void __exit(code);

int code;
```

Description

This function terminates the current program immediately and returns control to the parent program. This function does not write output buffers or close level 2 files. Generally, this function is used only in emergency situations when you do not care if some output data are lost.

Files opened with the open , creat , or creatx function are closed.

The code parameter is a value from 0 to 255 that gets passed back to the parent. By convention, a value of 0 indicates success.

```
  From within a shared library, you must not call any library
  functions that terminate your program.  For example, you cannot
  call  exit , __exit, or  abort  from a shared library.  You also
  cannot use  setjmp  and  longjmp  to jump across a call from the
  program into the library.
```

Portability

SAS/C

See Also

exit

1.122 exp()

exp-Exponential function

Synopsis

```
  #include <math.h>

  r = exp(x);

  double r, x;
```

Description

The exp function raises the natural logarithm base e to the x power. A range error occurs if x is too large.

Portability

ANSI

Returns

This function returns a double-precision floating-point number containing the calculated exponential.

See Also

log , __matherr

1.123 fabs()

fabs-Floating-point or double-precision floating-point absolute value

Synopsis

```
  #include <math.h>
  ad = fabs(d);

  double ad,d;
```

Description

```
  This function computes the absolute value of either a
  floating-point number or a double-precision floating-point number.
  Floating-point arguments are automatically promoted to
  double-precision floating-point arguments.
```

Portability

```
   ANSI
```

Returns

```
  This function returns a double-precision floating-point number
  containing the absolute value of the argument.
```

See Also

```
   abs
```

1.124 fclose()

fclose-Close a level 2 file

Synopsis

```
  #include <stdio.h>

  ret = fclose(fp);

  int ret;   /* return code */
  FILE *fp;  /* file pointer for file to be closed */
```

Description

```
  The fclose function completes the processing of a level 2 file and
  releases all related resources.  If the file is in the course of
  being written, any data which have accumulated in the buffer are
  written to the file, and the level 1 __close function is called
  for the associated file descriptor. The buffer associated with the
  file block is freed.

  Even though the fclose function is automatically called for all
  open files when your program terminates or calls the  exit
  function, it is good programming practice to close your own files
  explicitly. The last buffer is not written until the fclose
  function is called, and data may be lost if an output file is not
  properly closed.
```

Portability

```
   ANSI

Returns

  If successful, the fclose function returns 0.  This function
  returns EOF to indicate an error.  If EOF is returned, additional
  error information can be found in the external integers  errno  and
   _OSERR .

See Also

   errno ,  fopen ,  open ,  _OSERR
```

1.125 fcloseall()

```
fcloseall-Close all level 2 files

Synopsis

  #include <stdio.h>

  num = fcloseall();

  int num;  /* number of files closed */

Description

  The fcloseall function closes all level 2 files that were open and
  returns that number.  However, if an error occurs on any file, the
  fcloseall function continues to close the other files and then
  returns a value of -1.

  The fcloseall function closes the standard files stdin, stdout,
  and stderr.  Functions such as  printf  and  perror  will fail after
  you call the fcloseall function.

Portability

   XENIX

Returns

  If successful, the fcloseall function returns the number of files
  that were closed.  This function returns -1 to indicate an error.
  If -1 is returned, additional error information can be found in
  the external integers  errno  and  _OSERR .

See Also

   errno ,  fopen ,  _OSERR
```

1.126 fcvt()

```
fcvt-Convert a floating-point number to a string

Synopsis

  #include <math.h>

  s = fcvt(v,dig,decx,sign);

  char *s;   /* string pointer */
  double v;  /* floating point value */
  int dig;   /* number of digits */
  int *decx; /* pointer to decimal index (returned) */
  int *sign; /* pointer to sign indicator */

Description

  This function converts a floating-point number into an ASCII
  character string consisting of digits only and terminated by a
  null character.

  The second argument, dig, indicates the total number of digits
  that should be generated If the floating-point value contains
  fewer significant digits, zeroes are appended. If there are too
  many significant digits, the low-order (right-most) digit is
  rounded.

  The decx argument points to an integer that receives a value
  indicating where the decimal point should be placed in the string.
  For example, an index value of 3 indicates that the decimal point
  should be placed just after the third character in the string.  A
  value of 0 means that the decimal point is just before the first
  character. If the index is negative, it indicates the number of
  zeroes that are between the decimal point and the first character.
  For example, -3 means that there are three zeroes between the
  decimal point and the beginning of the string.

  The sign argument points to an integer that is nonzero if
  the value v is negative.

  This function is not available if the  _STRICT_ANSI  flag
  has been defined.

Portability

   UNIX

Returns

  This function returns a pointer to a string containing the ASCII
  representation of the float argument.

Example

  #include <stdio.h>
  #include <math.h>

  void main(void)
  {
      int decx, sign;
      char *string;

      string = fvct(3.1415926535,10,&decx,&sign);

      /* string => "3141592654" */
      /* decx   => 1          */
      /* sign   => 0          */

      printf("string = %s, decx = %d, sign = %d\n",
       string, decx, sign);
  }
```

See Also

   ecvt

1.127 fdopen()

fdopen-Attach a level 1 file to level 2

Synopsis

```
  #include <stdio.h>

  fp = fdopen(fh,mode);

  FILE *fp;     /*  file pointer */
  int fh;       /*  file handle */
  const char *mode;  /*  access mode */
```

Description

  This function attaches a level 1 file to level 2. In other words, if you have used the open function to prepare a file for level 1 I/O processing, you can subsequently use level 2 I/O with that file after calling the fdopen function. The file handle is the value returned by the open function, and the access mode has the same form as described for the fopen function.

Portability

   UNIX

Returns

  If the operation is successful, the function returns a file pointer other than NULL. If it is not successful, the function returns a NULL pointer and places error information in the external integers errno and _OSERR .

```
  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

See Also

```
   errno ,  fopen ,  _OSERR
```

1.128 feof()

feof-Check for a level 2 end-of-file

Synopsis

```
  #include <stdio.h>

  ret = feof(fp);

  int ret;   /* non-zero if condition is true */
  FILE *fp;  /* file pointer */
```

Description

```
  This function generates a nonzero value if the specified file
  pointer is at end-of-file.  This function is implemented as a
  macro.  Also, it does not check whether the fp argument is a valid
  file pointer.
```

Portability

```
   ANSI
```

Returns

```
  This function returns a nonzero value if the specified file
  pointer is at end-of-file.  If not, this function returns a 0.
```

See Also

```
   ferror
```

1.129 ferror()

ferror-Check for a level 2 error

Synopsis

```
  #include <stdio.h>

  ret = ferror(fp);

  int ret;   /* non-zero if condition is true */
  FILE *fp;  /* file pointer */
```

Description

```
  This function tests the error indicator for the file pointed to by
  the fp argument.  This function is implemented as a macro.  The
  ferror function does not check whether the fp argument is a valid
  file pointer.
```

Portability

```
   ANSI
```

Returns

```
  The return value is 0 if no error has been set.  If an error
  indicator was set, a nonzero value is returned.
```

See Also

```
   clearerr ,  feof
```

1.130 fflush()

fflush-Flush a level 2 output buffer

Synopsis

```
  #include <stdio.h>

  ret = fflush(fp);

  int ret;    /* return code */
  FILE *fp;   /* file pointer */
```

Description

```
  The fflush function flushes the output buffer of the specified
  level 2 file.  That is, it writes the buffer if the file is opened
  for output and the buffer contains any pending data.
```

Portability

```
   ANSI
```

Returns

```
  If an error occurs, the return value is -1 (EOF). The appropriate
  error code is placed into the external integer  errno , and
  additional information is placed in the external integer  _OSERR .
```

See Also

```
   errno ,  fclose ,  fcloseall ,  flushall ,  fopen ,  _OSERR
```

1.131 fgetc()

```
fgetc-Get a character from a file

Synopsis

  #include <stdio.h>

  c = fgetc(fp);

  int c;      /* return character or code */
  FILE *fp;   /* file pointer */

Description

  This function gets a single character from the specified level 2
  file.

  In the case of an error, this function returns an EOF.  To
  distinguish errors from an end-of-file condition, you should reset
  the external integer  errno  before calling the function, and
  analyze its contents when you receive an EOF return.

Portability

   ANSI

Returns

  If successful, the next input character is returned. Otherwise,
  the function returns EOF, which is defined in the file stdio.h.
  In the event of an EOF return, error information can be found in
  the external integers  errno  and  _OSERR .

See Also

   errno ,  fgetchar ,  fopen ,  fputc ,  getc ,  getch ,  getchar ,
   _OSERR ,  ungetc
```

1.132 fgetchar()

```
fgetchar-Get a character from stdin

Synopsis

  #include <stdio.h>

  c = fgetchar(void);

  int c;      /* return character or code */

Description

  This function gets a single character from stdin. In the case of
```

an error, this function returns an EOF. To distinguish errors from an end-of-file condition, you should reset the external integer errno before calling the function and analyze its contents when you receive an EOF return.

This function is identical to fgetc(stdin).

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

XENIX

Returns

If successful, the next input character is returned. Otherwise, the function returns EOF, which is defined in the stdio.h file. In the event of an EOF return, error information can be found in the external integers errno and _OSERR .

See Also

errno , fgetc , fopen , getc , getchar , _OSERR

1.133 fgetpos()

fgetpos-Get the current file position

Synopsis

```
#include <stdio.h>

x = fgetpos(f, pos);

int x;
FILE *f;
fpos_t *pos;
```

Description

The fgetpos function determines the current file position for the stream associated with the FILE object addressed by the f argument, and it stores the file position in the object pointed to by the pos argument. This object is of type fpos_t, which is defined in the stdio.h file. The stored value can be passed to the fsetpos function to reposition the file to its position at the time of the call to the fgetpos function.

The fgetpos function can be used with most types of files, using either text or binary access.

Portability

ANSI

Returns

If successful, the fgetpos function returns 0. If it fails, the fgetpos function returns a nonzero value and stores an appropriate error code in the external integer errno .

Example

In the following example, the function bldtable reads a file and builds a table of keys and record addresses. The function findrec positions the file to the record with the required key using the fsetpos function, and then it reads the record.

```
#include <stdio.h>
#include <string.h>

#define KEYLEN 17
#define DATALEN 500
#define TABSIZE 1000

struct table
{
    char keyval[KEYLEN];
    fpos_t location;
} keytable[TABSIZE];

struct record
{
    char keyval[KEYLEN];
    char data[DATALEN];
};

int filesize;

/* Initialize keytable, which is a */
/* table of keys and locations     */
void bldtable(FILE *fileptr)
{
    struct record input;
    int index =0;

    while (!feof(fileptr))
    {
  /* Store file pointer location */
  fgetpos(fileptr, &keytable[index].location);

  /* Read 1 record */
  fread(&input, sizeof(struct record), 1, fileptr);
  if (feof(fileptr) || ferror(fileptr))
      break;

  /* Save the keyval */
  memcpy(keytable[index].keyval, input.keyval, KEYLEN);
  index++;
    }
```

```
        filesize = index;
        return;
    }

    /* Find a match in the file to the key, */
    /* and return complete record.    */
         int findrec(FILE *fileptr, char keyval[KEYLEN], struct record *input)
         {
       int index;
       /* Search keytable for specified value */
       for (index = 0; index < filesize; ++index)
       if (memcmp(keyval, keytable[index].keyval, KEYLEN) == 0)
           break;

       if (index >= filesize)
           return (-1);  /* Keyval not found */

       /* If found, read complete record from file */
       fsetpos (fileptr, &keytable[index].location);
       fread (input, sizeof(struct record), 1, fileptr);
       return (0);
    }
```

See Also

fseek , fsetpos , ftell , lseek

1.134 fgets()

fgets-Get a string from a level 2 file

Synopsis

```
  #include <stdio.h>

  p = fgets(buffer,length,fp);

  char *p;       /* buffer pointer or NULL */
  char *buffer;  /* buffer pointer */
  int length;    /* buffer length in bytes */
  FILE *fp;      /* file pointer */
```

Description

The fgets function gets a string from the specified level 2 file, which must have been previously opened for input. Characters are copied from the file to the buffer until a new line (\n) has been copied, or length-1 characters have been copied, or the end-of-file is hit. In any case, if the read succeeds, the buffer is terminated with a trailing null byte (\0). If the read fails, the buffer will not be modified.

Portability

```
   ANSI
```

Returns

```
  If the end-of-file is hit before any bytes are read, a NULL
  pointer is returned.  If an I/O error occurs, a NULL pointer is
  returned and additional information is placed in the external
  integers  errno  and  _OSERR .  If no I/O error occurs and at least
  one byte was read from the file, the buffer argument is returned.
```

Example

```
  /* Assume that stdin contains the following lines: */
  /*       */
  /*  Hello, folks!  */
  /*  Goodbye, folks!  */

  #include <stdio.h>

  void main(void)
  {
      char *p,b[80];

      /* For the next two lines, p will point to b */
      p = fgets(b,sizeof(b),stdin);
      printf("b = %p, %sp = %p\n", b, b, p);
      /* b contains "Hello, folks!" */
      p = fgets(b,sizeof(b),stdin);
      printf("b = %p, %sp = %p\n", b, b, p);

      /* b now contains "Goodbye, folks!\n" */
      p = fgets(b,sizeof(b),stdin);
      printf("b = %p, %sp = %p\n", b, b, p);
  }
```

See Also

```
   errno ,  feof ,  ferror ,  fgetc ,  fopen ,  fread ,  getc ,  gets ,
   _OSERR
```

1.135 fileno()

fileno-Get the file number for a level 2 file

Synopsis

```
  #include <stdio.h>

  fh = fileno(fp);

  int fh;     /* file handle */
  FILE *fp;   /* file pointer */
```

Description

This function gets the file handle (the file number) associated with the specified file pointer. The file pointer must be one that was returned by the fopen , freopen , or fdopen function.

This function is implemented as a macro. Also, it does not check that the fp argument is a valid file pointer.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

UNIX

Returns

The return value is an integer representing the file handle.

See Also

fdopen , fopen , freopen

1.136 findpath()

findpath-Locate a file in the current path

Synopsis

```
#include <dos.h>

lock = findpath(filename);

BPTR lock;
const char *filename;
```

Description

This function locates a file in the currently defined path. If the process is not a Shell process, it uses the path in effect when Workbench was loaded.

Portability

AmigaDOS

Returns

If the findpath function finds the file, it returns a lock on that directory even if it is the current directory. The lock must be unlocked with the AmigaDOS UnLock function. If the findpath function cannot find the file, it returns a -1. The value NULL is not used because NULL is a valid lock.

Example

```
  #include <stdio.h>
  #include <stdlib.h>
  #include <dos.h>
  #include <proto/dos.h>
  #include <proto/exec.h>

  void main(int argc, char *argv[])
  {
      char path[256];
      long rc;
      BPTR lock;
      struct DosLibrary *DOSBase;

      rc = EXIT_FAILURE;

      if ((DOSBase = (struct DosLibrary *)
         OpenLibrary("dos.library", 0L)) == NULL)
      {
    printf("Couldn't open dos.library!\n");
      }
      else
      {
    if (argc < 2)
    {
        printf("You must enter a file to find!\n");
    }
    else
    {
        printf("Looking for \"%s\"... ", argv[1]);

        if ((lock = findpath(argv[1])) == ((BPTR)-1))
        {
      printf("Error!\nCannot find \"%s\" in "
             "the current path\n", argv[1]);
        }
        else
        {
      printf("Found it!\n");
      if (getpath(lock, path) == 0)
      {
          printf("Path: \"%s\"\n", path);
      }
      UnLock(lock);
      rc = EXIT_SUCCESS;
        }
    }
    CloseLibrary((struct Library *)DOSBase);
      }
      exit(rc);
  }
```

See Also

getpath , UnLock (The AmigaDOS Manual, 3rd Edition)

1.137 floor()

```
floor-Get the floor of a real number

Synopsis

  #include <math.h>

  x = floor(y);

  double x,y;

Description

  This function returns the largest integer not greater than the
  specified real number.

Portability

   ANSI

Returns

  The result is a real number.

Example

  #include <stdio.h>
  #include <math.h>

  void main(void)
  {
      double r;

      r = floor(523.96);   /* r contains 523.0 */
      printf("floor(523.96) = %lf\n", r);
  }

See Also

   ceil
```

1.138 flushall()

```
flushall-Flush all level 2 output buffers

Synopsis

  #include <stdio.h>

  num = flushall(void);

  int num;  /* number of open files */
```

```
Description

  The flushall function flushes all level 2 output buffers and
  returns the number of level 2 files that are open.  If an error
  occurs, the function continues to flush the remaining files and
  then returns a value of -1.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability
  XENIX
Returns

  If an error occurs, the return value is -1 (EOF).  The appropriate
  error code is placed into the external integer  errno , and
  additional information is placed in the external integer  _OSERR .

See Also

   errno ,  fclose ,  fcloseall ,  fflush ,  fopen ,  _OSERR
```

1.139 fmod()

```
fmod-Floating-point modulus operations

Synopsis

  #include <math.h>

  x = fmod(y,z);

  double x,y,z;

Description

  The fmod function calculates the floating-point remainder of y/z.
  If the % (modulus) operation were defined for floating-point
  numbers, the expression would produce the following:

      x = y % z;

Portability

   ANSI

Returns

  This function returns the value y if the value z is 0.  Otherwise,
  it returns a value that has the same sign as y, is less than z,
  and satisfies the following relationship:

      y = (i * z) + x

  The argument i is an integer.
```

Example

```
  #include <stdio.h>
  #include <math.h>

  void main(void)
  {
      double r;

      r = fmod(5.7,1.5);      /* r contains 1.2 */
      printf("fmod(5.7, 1.5) = %lf\n", r);
  }
```

See Also

modf

1.140 fmode()

fmode-Change the mode of a level 2 file

Synopsis

```
  #include <stdio.h>

  void fmode(fp,mode);

  FILE *fp;    /* file pointer */
  int mode;    /* 0 => mode A */
         /* 1 => mode B */
```

Description

This function is used to change the translation mode of a level 2 file that has been opened with the fopen , freopen , or fdopen function.

In mode A, carriage returns are deleted on input, and a carriage return is inserted before each line feed on output. Mode A also detects the Control-Z character (0x1A) as a logical end-of-file mark. In mode B, all data are transferred with no changes.

For AmigaDOS, the default mode is B.

The file pointer is not checked for validity.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

See Also

fdopen , fopen , freopen

1.141 fopen()

fopen-Open a level 2 file

Synopsis

```
  #include <stdio.h>

  fp = fopen(name,"mode");

  FILE *fp;       /* file pointer */
  const char *name;    /* file name */
  const char *mode;    /* access mode string */
```

Description

This function opens a file for buffered access. The name string can be any valid filename and may include a device code and directory path. The mode string specifies the values for each mode (Create, Trunc, Read, Write, Append, and Translate) with which you want to open the file.

NOTE: Do not place the const keyword on the declarations for the name and mode arguments in your program. For information about the const keyword, see the description of the Synopsis section at the beginning of this chapter.

The values of the modes Create, Trunc, Read, Write, Append, and Translate indicate how you want the file to be processed. The following table describes the effect of each setting of these modes.

```
      Mode     Value  Effect
      ----     -----  ------
      Create    yes   The file will be created if it does not
          already exist.
          no    The function will fail if the file does not
          already exist.
      Trunc     yes   If the file exists, it will be truncated
          (marked as empty).
          no    If the file exists, its current contents will
          not be disturbed.
      Read      yes   The file can be read with functions such as
           fread  and  fgetc .  Also, the  fseek  function
          can be used to position the file before
          reading.
          no    The file cannot be read.
      Write     yes   The file can be written with functions such as
           fwrite  and  fputc .  Also, the  fseek  function
          can be used to position the file before
          writing.
          no    The file cannot be written, but see Append
```

```
        below.
    Append    yes   The file can be written, but it is
        automatically positioned to the current
        end-of-file before each write operation.  This
        mode prevents existing data from being
        changed.
        no    Automatic positioning to the end-of-file is
        not done before a write operation.  Also,
        writes are not allowed unless the Write mode
        value is Yes.

Translate default

The external integer  __fmode  is used to set mode A or mode B as
follows:

    if (__fmode & 0x8000)
  set mode B
    else set mode A

For AmigaDOS, the external integer  __fmode  is normally 0x8000.

    mode A
  On a read operation, each carriage-return character (\r)
  is deleted, and the Control-Z character is treated as a
  logical end-of-file mark. On a write operation, each
  line-feed character (\n) is expanded to a carriage return
  followed by a line feed.

    mode B
  The data are unchanged as they are read or written.

The following table shows the list of values specified by each
mode string.
```

Mode String	Create	Trunc	Read	Write	Append	Translate
r	no	no	yes	no	no	default
w	yes	yes	no	yes	no	default
a	yes	no	no	no	yes	default
r+	no	no	yes	yes	no	default
w+	yes	yes	yes	yes	no	default
a+	yes	no	yes	no	yes	default
ra	no	no	yes	no	no	mode A
wa	yes	yes	no	yes	no	mode A
aa	yes	no	no	no	yes	mode A
ra+	no	no	yes	yes	no	mode A
wa+	yes	yes	yes	yes	no	mode A
aa+	yes	no	yes	no	yes	mode A
rb	no	no	yes	no	no	mode B
wb	yes	yes	no	yes	no	mode B
ab	yes	no	no	no	yes	mode B
rb+	no	no	yes	yes	no	mode B
wb+	yes	yes	yes	yes	no	mode B
ab+	yes	no	yes	no	yes	mode B

If the file is successfully opened, the function returns a pointer to a buffered I/O control block, which is defined in the header file stdio.h. Normally, you will not need to access any information in the control block directly, but you should be very careful not to disturb the block accidentally. A common C programming error is to accidentally mutilate one of these control blocks, which can cause garbage to be written into a file.

Portability

ANSI

Mode A open modes are an extension to the ANSI standard, and you should not use them in programs that you want to be ANSI-compatible.

Returns

A NULL pointer is returned if the file cannot be opened. Consult the external integers errno and _OSERR for detailed error information.

When a file is opened for both reading and writing, you should call the fseek or rewind function when switching from reading to writing or vice-versa. It is not necessary to do this when you begin writing after reading up to the end of the file.

Example

```
/* This is an example of using fopen to write a copy */
/* file routine.  It returns 0 if the file copy was  */
/* successful; otherwise, it returns a -1.       */

#include <stdio.h>

int copy (char *infile, char *outfile)
{
    FILE *in,*out;
    char buf[100];
    int i;

    /* open the input file */
    if ((in=fopen(infile,"r"))==NULL)
  return(-1);
    if ((out=fopen(outfile,"wt"))==NULL)
    {
  fclose(in);
  return(-1);
    }

    /* copy the file contents */
    while (i=fread(buf,1,100,in))
  if (fwrite(buf,1,i,out)!=i)
      break;

    /* now set up the return */
    i=(ferror(in)||ferror(out))?-1:0;
```

```
      /* close the files */
      fclose(in);
      fclose(out);
      return(i);
  }
```

See Also

   fclose , fdopen , fgetc , fgets , fputc , fputs , fread ,
   freopen , fwrite , open

1.142 forkl()

forkl-Create a child process with an argument list

Synopsis

```
  #include <dos.h>

  error = forkl(prog,arg0,arg1,...,argn,NULL,env,procid);

  int error;     /* error code */
  char *prog;    /* program name */
  char *arg0;    /* argument #0 */
  char *arg1;    /* argument #1 */
  char *argn;    /* argument #n */

  struct FORKENV *env;    /* pointer to pseudo environment*/
          /* structure (may be NULL)*/
  struct ProcID *procid;    /* pointer to process ID structure*/
```

Description

The forkl function creates a child process by loading a new program as a concurrent process. The parent continues to execute until it calls either the wait or waitm function; that is, the parent and child are multiprogrammed. When the child process completes, the parent process (the current program) can get its completion code with the wait or waitm function.

To specify the arguments for the new program with the forkl function, specify a list of argument string pointers terminated by a NULL pointer (arg0,arg1,...,argn,NULL). Following UNIX conventions, the first argument (arg0) should be the program name and is normally the same as the prog argument. Under AmigaDOS, the arguments are all concatenated into a pseudo-command line, with a blank separating adjacent arguments and a carriage-return character at the end. The maximum size of this line is 255 bytes.

The pseudo-environment pointer env, if specified, contains optional data about default files and process execution. The FORKENV structure is defined in dos.h.

The child process is supplied with default values for any field left null. In addition, a NULL pointer may be passed for this parameter, which causes default values to be used for all items. If the default values are used, the child process is created with a priority of 0, a stack size of 8000 bytes, the current stdin and stdout files, and console for the parent process, and a new message port is created to receive the termination message.

The new process executes as a CLI-type task. This means it will be expecting file handles for stdin and stdout to be present, and a console task handler to exist. If the parent process is running as a Workbench process then it is possible for none of these to exist for the child process to inherit. If the parent process can be invoked from Workbench, extra effort should be made to ensure the presence of file handles the child may require. These file handles are BPTR values, as returned by the AmigaDOS Open call.

The optional message-port field is provided to allow the parent process to detect when a child process has terminated while awaiting other events, such as Intuition menu events. The termination message format is similar to an Intuition message. The TermMsg structure is defined in dos.h.

When a termination message is received, the parent process must still call the wait function to remove the message and unload the child process. If the message was removed from the port, it must be replaced by calling the PutMsg function before calling the wait function.

The forkl function loads the program from the current path using the AmigaDOS search procedure. If the calling program was loaded from the Workbench, the path used is the path at the time the Workbench was loaded. Alternatively, you can include a path specification as part of the program name.

Upon successful creation of the child process, the forkl function fills in the process ID structure. The address of this structure must be passed as the last argument. The ProcID structure is defined in dos.h.

The nextID field may be used to link process ID structures into a simple linked list for use with the waitm function. The process field is the address of the process's task structure and is the value used with ROM Kernel functions, such as the signal function, that require a task ID parameter. The remaining fields are used by the wait function.

The wait or waitm function must be called for each child process to ensure that the child process terminates cleanly. The wait function takes as its single argument a pointer to the ProcID structure for the child task. It waits for a termination message from one particular process, replying to and discarding any other messages that arrive at the message port. The function returns the child process's completion code. This is the value that was passed to the exit function when the child terminated.

If any child processes share a message port, however, then the waitm function should be called to ensure that no termination messages are lost. The waitm function requires the address of a pointer to the first ProcID structure in a linked list of child process ProcID structures. It waits for one or more termination messages, removing the ProcID structure from the original linked list and inserting it into a linked list of terminated process ProcID structures. The completion code is placed in the UserPortFlag field of the structure. The function then returns a pointer to the first structure in the list of terminated process structures. Since the original list is updated, it may be reused to wait for the remaining child processes.

The wait or waitm function must be called for each child process before terminating the parent process. Otherwise, the child process will never be unloaded, and there is an excellent possibility that the system will crash.

NOTE: BCPL programs cannot be executed using the forkl function. These programs include all of the AmigaDOS routines under Workbench 1.3. The AmigaDOS routines from Workbench 2.0 will fork properly.

Portability

SAS/C

Returns

If the specified program file cannot be found, a -1 return is made, and additional error information can be found in the external integers errno and _OSERR . You must call the wait function to obtain the completion code from the child process.

Example

```
/* This example forks multiple child processes */

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>

void main(void)
{
    struct ProcID *children, *terminated, *task;
    struct ProcID child1, child2, child3;
    int taskno;

    if (forkl("task1","task1","argument1","argument2",
        NULL,&child1) == -1)
    {
  printf("error forking child1\n");
  exit(EXIT_FAILURE);
    }

    if (forkl("task2","task2","argument1","argument2",
        NULL,&child2) == -1)
    {
  printf("error forking child2\n");
```

```
    exit(EXIT_FAILURE);
      }

      if (forkl("task3","task3","argument1","argument2",
          NULL,&child3) == -1)
      {
    printf("error forking child3\n");
    exit(EXIT_FAILURE);
      }

      child3.nextID = NULL;
      child2.nextID = &child3;
      child1.nextID = &child2;

      children = &child1;

      while(children)      /* wait until no more children */
      {
    /* must pass ADDRESS of pointer */
    terminated = waitm(&children);
    for (task = terminated; task != NULL; task = task->nextID)
    {
        if (task == &child1)
        {
      taskno = 1;
        }
        else if (task == &child2)
        {
      taskno = 2;
        }
        else if (task == &child3)
        {
      taskno = 3;
        }
        printf("task %d terminated, value = %d\n",
         taskno,task->UserPortFlag);
    }
      }
  }
```

See Also

```
   exit ,  forkv ,  wait ,  waitm ; LoadSeg, CreateProc,
  Execute, System, and Open in The AmigaDOS Manual, 3rd Edition
```

1.143 forkv()

forkv-Create a child process with an argument vector

Synopsis

```
  #include <dos.h>

  error = forkv(prog,argv,env,procid);
  int error;    /* error code */
  char *prog;   /* program name */
  char *argv[];   /* argument vector */

  struct FORKENV *env;    /* pointer to pseudo environment*/
          /* structure (may be NULL)*/
  struct ProcID *procid;    /* pointer to process ID structure*/
```

Description

The forkv function creates a child process by loading a new program as a concurrent process. The parent continues to execute until it calls either the wait or waitm function; that is, the parent and child are multiprogrammed. When the child process completes, the parent process (the current program) can get its completion code with the wait or waitm function.

To specify the arguments with the forkv function, specify a single pointer (argv) to an array of argument string pointers, with the array being terminated by a NULL pointer. Following UNIX conventions, the first argument (argv[0]) should be the program name and is normally the same as the prog argument. Under AmigaDOS, the arguments are all concatenated into a pseudo-command line, with a blank separating adjacent arguments and a carriage-return character at the end. The maximum size of this line is 255 bytes.

See the description of the forkl function for more details.

NOTE: BCPL programs cannot be executed using the forkv function. These programs include all of the AmigaDOS routines under Workbench 1.3. The AmigaDOS routines from Workbench 2.0 will fork properly.

Portability

SAS/C

Returns

If the specified program file cannot be found, a -1 return is made, and additional error information can be found in the external integers errno and _OSERR . You must call the wait function to obtain the completion code from the child process.

Example

The following program, child.c, creates a child process.

```
  /* This program creates a child process and displays */
  /* the return code.  The child program name and      */
  /* arguments are taken from the command line.      */

  #include <stdio.h>
  #include <stdlib.h>
  #include <dos.h>
```

```
struct ProcID child;

void main(int argc, char *argv[])
{
    int ret;
    if (argc < 2)
    {
  printf("no program specified\n");
  printf("usage: fork program [arg1] ... [argn]\n"
  exit(EXIT_FAILURE);
    }

    printf("parent: beginning fork of %s\n",argv[1]);
    if (forkv(argv[1],&argv[1],NULL,&child) == -1)
    {
  printf("error forking child\n");
  exit(EXIT_FAILURE);
    }
    else
    {
  ret = wait(&child);
    }
    printf("parent: %s finished, ret = %d\n",argv[1],ret);
}
```

The following program, parent.c, forks multiple child processes.

```
/* This example forks multiple child processes */

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>

char *child1_argv[] ={"task1",          /* program name */
          "argument1",      /* 1st argument */
          "argument2",      /* etc., etc.   */
          NULL};
char *child2_argv[] ={"task2",          /* program name */
          "argument1",      /* 1st argument */
          "argument2",      /* etc., etc.   */
          NULL};

char *child3_argv[] ={"task3",          /* program name */
          "argument1",      /* 1st argument */
          "argument2",      /* etc., etc.   */
          NULL};

void main(void)
{
    struct ProcID *children, *terminated, *task;
    struct ProcID child1, child2, child3;
    int taskno;

    if (forkv(child1_argv[0],child1_argv,NULL,&child1) == -1)
    {
  printf("error forking child1\n");
  exit(EXIT_FAILURE);
```

```
      }

      if (forkv(child2_argv[0],child2_argv,NULL,&child2) == -1)
      {
    printf("error forking child2\n");
    exit(EXIT_FAILURE);
      }

      if (forkv(child3_argv[0],child3_argv,NULL,&child3) == -1)
      {
    printf("error forking child3\n");
    exit(EXIT_FAILURE);
      }

      child3.nextID = NULL;
      child2.nextID = &child3;
      child1.nextID = &child2;

      children = &child1;

      while(children)      /* wait until no more children */
      {
    /* must pass ADDRESS of pointer */
    terminated = waitm(&children);
    for (task = terminated; task != NULL; task = task->nextID)
    {
        if (task == &child1)
        {
      taskno = 1;
        }
        else if (task == &child2)
        {
      taskno = 2;
        }
        else if (task == &child3)
        {
      taskno = 3;
        }
        printf("task %d terminated, value = %d\n",
         taskno,task->UserPortFlag);
    }
      }
  }
```

See Also

   exit , wait , waitm ; LoadSeg, CreateProc, Execute, System, and
  Open in The AmigaDOS Manual, 3rd Edition

1.144 fprintf()

fprintf-Formatted print

Synopsis

```
  #include <stdio.h>

  length = fprintf(fp,fmt,arg1,arg2,...);

  int length;       /* number of characters generated */
  FILE *fp;       /* file pointer */
  const char *fmt;      /* format string */
  type *argn;       /* arguments */

Description

  This function produces an output stream of ASCII characters, and
  sends the output to the level 2 file specified by the fp argument.

  The fmt argument points to a string that contains ordinary
  characters and conversion specifications that indicate how you
  want the arguments arg1, arg2, and so on to be printed.  The
  ordinary characters are copied to the output, but the conversion
  specifications are replaced with the correctly formatted values of
  the arguments arg1, arg2, and so on. The first conversion
  specification is replaced with the formatted value of arg1, the
  second specification is replaced with the  value of arg2, and so
  on.  In some cases, as described below, a conversion specification
  may process more than one argument.

  Each conversion specification must begin with a percent character
  (%).  To place an ordinary percent into the output stream, precede
  it with another percent in the fmt string.  That is, %% will send
  a single percent character to the output stream.  A specification
  has the following format:

      %[flags][width][.precision][size]type

  The brackets ([]) indicate optional fields.  Each field is defined
  as follows:

      flags
    controls output justification and the printing of signs,
    blanks, decimal places, and hexadecimal prefixes.

    If any flag characters are used, they must appear after
    the percent.  Valid flags are as follows:

    - (minus)
        causes the result to be left-justified within the
        field specified by width or within the default width.

    + (plus)
        causes a plus or minus sign to be placed before the
        result.  This flag is used in conjunction with the
        various numeric conversion types.  If it is absent,
        the sign character is generated only for a negative
        number.

    blank
        causes a leading blank for a positive number and a
        minus sign for a negative number.  This flag is
        similar to the plus.  If both the plus and the blank
        flags are present, the plus takes precedence.

    # (pound)
        causes special formatting.  With the o, x, and X
        types, the pound flag prefixes any nonzero output with
        0, 0x, or 0X, respectively.  With the f, e, and E
        conversion types, the pound flag forces the result to
        contain a decimal point.  With the g and G types, the
        pound flag forces the result to contain a decimal
        point and retain trailing zeroes.

    0 (zero)
        pads the field width with leading zeros instead of
        spaces for the d, i, o, u, x, X, e, E, f, g, and G
        conversion types.  If the minus flag is also used, the
        zero flag is ignored.  If a precision is specified,
        the zero flag is ignored for conversion types d, i, o,
        u, x, and X.  Behavior of the zero flag is undefined
        for the remaining conversion types.

      width
    specifies the field width, which is the minimum number of
    characters to be generated for this format item.

    The width is a nonnegative number that specifies the
    minimum field width.  If fewer characters are generated by
    the conversion operation, the result is padded on the left
    or right (depending on the minus flag described above).  A
    blank is used as the padding character unless width begins
    with a zero.  In that case, zero padding is performed.  If
    the minus flag appears, padding is performed with blanks.
    width specifies the minimum field width, and it will not
    cause lengthy output to be truncated.  Use the precision
    specifier for that purpose.

    If you do not want to specify the field width as a
    constant in the format string, you can code it as an
    asterisk (*), with or without a leading zero.  The
    asterisk indicates that the width value is an integer in
    the argument list.  See the examples for more information
    on this technique.

      precision
    specifies the field precision, which is the required
    precision of numeric conversions or the maximum number of
    characters to be copied from a string, depending on the
    type field.

    The meaning of the precision item depends on the field
    type, as follows:

        Type          Meaning
        ----          -------
         c            The precision item is ignored.
         d, i, o, u, x, X  The precision is the minimum number
               of digits to appear.  If fewer
```

```
           digits are generated, leading
           zeroes are supplied.
     e, E, f        The precision is the number of
           digits to appear after the decimal
           point.  If fewer digits are
           generated, trailing zeroes are
           supplied.
     g, G        The precision is the maximum number
           of significant digits.
     s           The precision is the maximum number
           of characters to be copied from the
           string.

 As with the width item, you can use an asterisk for the
 precision to indicate that the value should be picked up
 from the next argument.

   size
 can be either L for long double, l for large size, or h
 for small size.  When used with the d, i, o, u, x, or X
 conversion specifiers, h and l select argument types of
 short * and long *, respectively.  When used with the e,
 E, f, g, or G conversion specifiers, the L specifies a
 long double argument instead of a double.

   type
 specifies the type of argument conversion to be done.
 Valid conversion types are as follows:

     c
   specifies single-character conversion.  The
   associated argument must be an integer.  The
   single character in the right-most byte of the
   integer is copied to the output.
     d
   specifies decimal-integer conversion.  The
   associated argument must be an integer, and the
   result is a string of digits preceded by a sign.
   If the plus and blank flags are absent, the sign
   is produced only for a negative integer.  If the
   large size modifier is present, the argument is
   taken as a long integer.
     e
   specifies double-precision floating-point
   conversion.  The associated argument must be a
   double-precision floating-point number, and the
   result has the form:

       -d.dde-ddd

   d is a single decimal digit, dd is one or more
   digits, and ddd is an exponent of at least two
   digits.  The first minus sign is omitted if the
   floating-point number is positive, and the second
   minus sign is omitted if the exponent is positive.
   The plus and blank flags dictate whether there
   will be a sign character emitted if the number is
```

positive. The number of digits before the decimal point depends on the magnitude of the number, and the number after the decimal point depends on the requested precision. The value is rounded to the specified number of digits. If no precision is specified, the default is six decimal places.

E

specifies double-precision floating-point conversion. This is exactly the same as type e except that the result has the form:

```
-d.ddE-ddd
```

f

specifies double-precision floating-point conversion. The associated argument must be a double-precision floating-point number, and the result has the form:

```
-dd.dd
```

dd indicates one or more decimal digits. The minus sign is omitted if the number is positive, but a sign character will still be generated if the plus or blank flag is present. The number of digits before the decimal point depends on the magnitude of the number, and the number after the decimal point depends on the requested precision. If no precision is specified, the default is six decimal places. If the precision is specified as 0, or if there are no nonzero digits to the right of the decimal point, then the decimal point is omitted unless the pound (#) flag is specified.

g

specifies double-precision floating-point conversion (general form). The associated argument must be a double-precision floating-point number, and the result is in the e or f format, depending on which gives the most compact result. The e format is used only when the exponent is less than -4 or greater than the specified or default precision. Trailing zeroes are eliminated, and the decimal point appears only if any nonzero digits follow it.

G

specifies double-precision floating-point conversion (general form). This is identical to the g format, except that the E type is used instead of the e type.

i

specifies decimal-integer conversion. The associated argument must be an integer, and the result is a string of digits preceded by a sign. If the plus and blank flags are absent, the sign is produced only for a negative integer. If the large size modifier is present, the argument is taken as a long integer.

n

```
      specifies the argument will be a pointer to an
      integer into which is written the number of
      characters written so far by this call to the
      fprintf function.  If the large size flag is on,
      the argument must be a pointer to a long integer.
      If the small size flag is on, the argument must be
      a pointer to a short integer.
        o
      specifies octal-integer conversion.  The
      associated argument is taken as an unsigned
      integer, and it is converted to a string of octal
      digits. If the large size modifier is present, the
      argument must be a long integer.
        p
      specifies pointer conversion.  The associated
      argument is taken as a data pointer, and it is
      converted to hexadecimal representation.  Under
      AmigaDOS, the pointer is printed as 8 hexadecimal
      digits, with leading zeroes if necessary.
        P
      specifies pointer conversion.  This is the same as
      the p format, except that uppercase letters are
      used as hexadecimal digits.  This conversion type
      is an extension to the ANSI standard.  Do not use
      this extension if you want your program to be
      ANSI-compatible.
        s
      specifies string conversion.  The associated
      argument must point to a null-terminated character
      string.  The string is copied to the output, but
      the null byte is not copied.
        u
      specifies unsigned decimal integer conversion.
      The associated argument is taken as an unsigned
      integer, and it is converted to a string of
      decimal digits.  If the large size modifier is
      present, the argument must be a long integer.
        x
      specifies hexadecimal-integer conversion.  The
      associated argument is taken as an unsigned
      integer, and it is converted to a string of
      hexadecimal digits with lowercase letters.  If the
      large size modifier is present, the argument is
      taken as a long integer.
        X
      specifies hexadecimal-integer conversion.  This is
      the same as the x format, except that uppercase
      letters are used as hexadecimal digits.
```

Portability

   ANSI

Returns

  This function returns the number of output characters generated.
  If an error occurs, the fprintf function returns a negative value

```
  and places additional information in the external integers  errno
  and  _OSERR .
```

Example

```
  /* This example prints a message indicating whether */
  /* the function argument is positive or negative.   */
  /* In the second printf, the width and precision    */
  /* are 15 and 8, respectively.          */

  #include <stdio.h>
  #include <math.h>

  void pneg(double value)
  {
      char *sign;

      if (value < 0)
      {
    sign = "negative";
      }
      else
      {
    sign = "not negative";
      }
      fprintf(stdout,"The number %E is %s.\n",value,sign);
      fprintf(stdout,"The number %*.*E is %s.\n",15,8,value,sign);
  }

  void main(void)
  {
      pneg(37.8);
      pneg(-18.2);
  }
```

See Also

```
   errno ,  fscanf ,  _OSERR ,  printf ,  scanf ,  sprintf ,  sscanf
```

1.145 fputc()

fputc-Put a character to a level 2 file

Synopsis

```
  #include <stdio.h>

  r = fputc(c,fp);

  int r;      /* EOF or c */
  int c;      /* character to be output */
  FILE *fp;   /* level 2 file pointer */
```

Description

This function puts a single character to the specified level 2 file.

Portability

ANSI

Returns

If successful, this function returns the character c; otherwise, it returns EOF. For disk files, an EOF return usually means that the disk is full. However, this type of return can also occur if the device is write-protected or if a write error occurs. In any case, additional error information can be found in the external integers errno and _OSERR .

See Also

errno , fopen , fputchar , _OSERR , putc , putchar

1.146 fputchar()

fputchar-Put a character to stdout

Synopsis

```
  #include <stdio.h>

  r = fputchar(c);

  int r;      /* EOF or c */
  int c;      /* Character to be output */
```

Description

This function puts a single character to stdout.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

XENIX

Returns

If successful, this function returns the character c; otherwise, it returns EOF. For disk files, an EOF return usually means that the disk is full. However, this type of return can also occur if the device is write-protected or if a write error occurs. In any case, additional error information can be found in the external integers errno and _OSERR .

See Also

errno , fopen , fputc , _OSERR , putc , putchar

1.147 fputs()

fputs-Put a string to a level 2 file

Synopsis

```
  #include <stdio.h>

  error = fputs(s,fp);

  int error;    /* non-zero if error */
  const char *s;    /* string pointer */
  FILE *fp;   /* file pointer */
```

Description

The fputs function writes the string s to a level 2 file that was previously opened for output. The string must be terminated by a null byte, which is not written.

Portability

ANSI

Returns

If an error occurs, the return value is EOF; otherwise, it is 0. Additional error information can be found in the external integers errno and _OSERR .

Example

```
  /* This example writes the following two lines to stdout: */
  /*                */
  /* This is the first line         */
  /* This is the second line

  #include <stdio.h>

  void main(void)
  {
      puts("This is the first line");
      fputs("This is ",stdout);
      puts("the second line");
  }
```

See Also

errno , ferror , fopen , fputc , puts

1.148 fqsort()

```
fqsort-Sort an array of floating-point numbers

Synopsis

  #include <stdlib.h>

  void fqsort(fa,n);

  float *fa;   /* pointer to float array */
  stze_t n;    /* number of elements in array */

Description

  The fqsort function sorts the specified array of floating-point
  numbers using the ACM 271 algorithm, more popularly known as
  Quicksort.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

See Also

   dqsort ,  lqsort ,  qsort ,  sqsort ,  tqsort
```

1.149 fread()

```
fread-Read and write blocks

Synopsis

  #include <stdio.h>

  a = fread(b,bsize,n,fp);

  size_t a;      /* actual number of blocks */
  void *b;       /* pointer to first block */
  size_t bsize;  /* size of block in bytes */
  size_t n;      /* maximum number of blocks */
  FILE *fp;      /* file pointer */

Description

  This function uses level 2 I/O operations to read blocks of data.
  Each block contains bsize bytes and up to n blocks are stored into
  contiguous memory locations beginning at location b.

  Blocks are read until n blocks have been stored or until the
  end-of-file is hit.  If the end-of-file is hit in the middle of a
```

block, that partial block will be stored in the b array, but it will not be included in the function return value. In other words, the return value indicates the number of complete blocks that were read.

Portability

ANSI

Returns

This function returns the number of complete blocks read.

See Also

fclose , feof , ferror , fgetc , fopen , fputc , fseek , fwrite

1.150 free()

free-Free memory

Synopsis

```
#include <stdlib.h>

void free(b);

void *b;   /* block pointer */
```

Description

The free function releases a block of memory that was previously obtained using the calloc , malloc , or realloc function. For compatibility with some versions of UNIX, the block is not actually returned to the free space pool until the next time you call the calloc , malloc , realloc , or free function. Then, if that next call is to the realloc function and the block being reallocated is the one that was just freed, the realloc function will proceed correctly. In other words, you can ask the realloc function to reallocate a block that was freed as long as you have not called the calloc , malloc , or realloc function in the meantime.

Portability

ANSI

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct LIST
```

```
{
    struct LIST *next;
    char text[2];
};

void main(int argc, char *argv[])
{
    struct LIST *p;
    struct LIST *q;
    struct LIST list;
    char b[256];
    int x;

    printf("\nBegin new group...\n");
    for (q = &list; ; q = p)
    {
  printf("Enter a text string: ");
  if (fgets(b,sizeof(b),stdin) == NULL)
  {
      break;
  }
  if (b[0] == NULL)
  {
      if (q == &list)
      {
    printf("\n");
    exit(EXIT_SUCCESS);
      }
      break;
  }
  x = sizeof(struct LIST) - 2 + strlen(b) + 1;
  p = (struct LIST *)malloc(x);
  if (p == NULL)
  {
      printf("No more memory\n");
      exit(EXIT_FAILURE);
  }
  q->next = p;
  p->next = NULL;
  strcpy(p->text, b);
    }

    printf("\n\nTEXT LIST...\n");
    /*
     * You must be sure to copy the next pointer from
     * the current block before you free it.  Some
     * systems rely on a side-effect to be able to
     * access the memory after it is freed -- this is
     * BAD PROGRAMMING PRACTICE!
     */
    p = list.next;
    while(p != NULL)
    {
  q = p->next;
  printf(p->text);
  free((char *)p);
  p = q;
```

```
      }
      list.next = NULL;
  }
```

See Also

```
   calloc ,  getmem ,  malloc ,  rbrk ,  realloc ,  rlsmem ,  sbrk
```

1.151 freopen()

freopen-Reopen a level 2 file

Synopsis

```
  #include <stdio.h>

  fpr = freopen(name, mode, fp);

  FILE *fpr;       /* file pointer after re-opening */
  const char *name;    /* file name */
  const char *mode;    /* access mode */
  FILE *fp;      /* current file pointer */
```

Description

  This function reopens a level 2 file. That is, it attaches a new
  file to a previously used file pointer. The previous file is
  automatically closed before the file pointer is reused. The name
  and mode arguments are the same as those for the fopen function.

Portability

   ANSI

Returns

  If successful, this function returns the file pointer.

  Check the return code for the value NULL; the same errors as
  defined for the fopen function may occur. Also, for complete
  portability, do not assume that the fpr and fp pointers are
  identical. Use the fpr pointer to access the reopened file, not
  the fp pointer.

See Also

```
   fdopen ,  fopen
```

1.152 frexp()

frexp-Split floating-point value

```
Synopsis

  #include <math.h>

  f = frexp(v,xp);

  double f;    /* fraction */
  double v;    /* value */
  int *xp;     /* exponent pointer */

Description

  The frexp function splits the floating-point value v into its
  fraction (mantissa) and exponent parts.

Portability

   ANSI

Returns

  This function returns the mantissa as a double-precision
  floating-point number whose absolute value is greater than or
  equal to 0.5 and less than 1.0.  The exponent is returned as an
  integer whose absolute value is less than 1024.  If the value v is
  0, both returned values will be 0.

See Also

   fmod ,  ldexp ,  __matherr ,  modf
```

1.153 fscanf()

```
fscanf-Formatted input conversions

Synopsis

  #include <stdio.h>

  n = fscanf(fp,fmt,arg1,arg2,...);

  int n;      /* number of input items matched, or EOF */
  FILE *fp;   /* file pointer (fscanf only) */
  const char *fmt;  /* format string */
  type *argx;    /* pointers to input data areas */

Description

  This function reads formatted input from the specified level 2
  file, fp.  The input characters are read and checked against the
  format string, which may contain any of the following:

      white space
    Any number of spaces, horizontal tabs, or new-line
    characters cause input to be read up to the next character
    that is not white space.

      ordinary characters
    Any character that is not white space and is not the
    percent sign (%) must match the next input character.  If
    there is not an exact match, scanning stops, and the
    function returns.

      conversion specification
    This is a multicharacter sequence that indicates how the
    next input characters are to be converted.  The following
    paragraphs describe this conversion specification.

  The conversion specification follows this format, where brackets
  ([]) indicate an optional part:

      %[*][n][l|h]t

  The various fields are defined as follows:

      %
    introduces a conversion specifier.  If you want to match a
    percent sign in the input, use a double percent (%%) in
    the format string.

      *
    means that the conversion should be performed, but the
    result should not be stored.  You should not specify a
    pointer for any conversion specification that uses the
    asterisk (*) to suppress conversion.

      n
    specifies the maximum input field width and should be a
    decimal number.  This is used only with the s format.

      l
    indicates that a long integer conversion should be
    performed.  If neither l nor h is specified, the default
    is an integer.

      h
    indicates that a short conversion should be performed.  If
    neither l nor h is specified, the default is an integer.

      t
    stands for one of the following format characters:

        c, d, e, f, g, i, n, o, s, u, x, and [].

    These characters specify how the input characters are to
    be converted.

  The following list describes each of the format characters.

      c
    specifies character conversion.  The corresponding
    argument must point to a character.  The next input
```

character is moved to that destination. No white space is skipped.

d
specifies decimal number conversion. The corresponding argument must point to an integer or to a long integer if the d is preceded by an l. The input characters should be decimal digits and may be preceded by a plus or minus sign.

e,f,g
specifies floating-point conversion. These three types are identical. The corresponding argument must point to a floating-point number or to a double-precision floating-point number if the type letter is preceded by an l. The input characters must follow this format, where brackets ([]) indicate an optional part:

```
[whitespace][sign]digits[.digits][exponent]
```

- leading white space
- a plus (+) or minus (-) sign
- a sequence of decimal digits
- a decimal point followed by 0 or more decimal digit
- an exponent, consisting of the letter e or E followed by an optional plus or minus sign followed by one or more decimal digits

n
indicates a character count. No input characters are read. The corresponding argument must point to an integer into which is written the number of input characters read so far.

o
indicates an octal number. The corresponding argument should point to an integer, or to a long integer if the o is preceded by an l.

s
indicates a character string. The corresponding argument should point to a character array large enough to hold the string and a terminating null byte. The input string is terminated by white space or the end-of-input. Also, if a maximum field width is specified, the output array size should be at least that width plus 1 because the reading of input characters will stop at the field width even if no white space has been encountered.

u
indicates an unsigned integer. The corresponding argument should point to an unsigned integer or to an unsigned long integer if the u is preceded by an l.

```
      x
    indicates a hexadecimal integer.  The corresponding
    argument should point to an integer or to a long integer
    if the x is preceded by an l.  The hexadecimal number can
    begin with the characters 0x or 0X, and case is not
    significant for the hexadecimal letters.

      []
    indicates a string comprised of a specific set of
    characters.  A terminating null character is automatically
    added.  The corresponding argument should point to an
    array large enough to hold the sequence plus the
    terminating null character.

  Except for the c and [] specifiers, white space characters in the
  format string cause white space characters in the input to be
  skipped.

  If the conversion is successful and the assignment is not
  suppressed, the result is placed into the corresponding argument.
  The argument list must contain a pointer to an appropriate data
  item for each conversion specification that does not suppress
  assignment.

Portability

   ANSI

Returns

  The function returns the number of assignments that were made.
  For example, a return value of 3 indicates that conversion results
  were assigned to the arguments arg1, arg2, and arg3.  The number
  of assignments can be less than the number expected if the input
  characters do not agree with the format string.  If an
  end-of-input is reached before any values are assigned, the return
  value is EOF.

See Also

   scanf ,  sscanf
```

1.154 fseek()

```
fseek-Set a level 2 file position

Synopsis

  #include <stdio.h>

  error = fseek(fp,rpos,mode);

  int error;  /* non-zero if error */
  FILE *fp; /* file pointer returned from fopen() */
  long int rpos;  /* relative file position */
  int mode; /* seek mode */
```

Description

  The fseek function moves the byte cursor of a level 2 file to a
  new position. The mode argument must be one of the following:

      SEEK_SET
    seek from the beginning of the file. The rpos argument is
    the number of bytes from the beginning of the file. This
    value must be positive.

      SEEK_CUR
    seek from the file's current position. The rpos argument
    is the number of bytes relative to the current position.
    This value can be positive or negative.

      SEEK_END
    seek from the end of the file. The rpos argument is the
    number of bytes relative to the end of the file. This
    value must be negative or 0.

Portability

   ANSI

Returns

  A value of -1 is returned if an error occurs. The external
  integers errno and _OSERR contain additional error information.

See Also

   errno , fopen , ftell , lseek , _OSERR , rewind

1.155 fsetpos()

fsetpos-Reposition a file

Synopsis

```
  #include <stdio.h>

  x = fsetpos(fp,pos);

  int x;
  FILE *fp;
  const fpos_t *pos;
```

Description

  The fsetpos function positions the file pointed to by the fp
  argument to the position specified by the object pointed to by the
  pos argument. This object is of type fpos_t, which is defined in

the stdio.h file.

The value of the object pointed to by the pos argument should be set by a previous call to the fgetpos function for the same file.

The fsetpos function can be used with most files, accessed either as text or binary. The fsetpos function clears the EOF indicator for the file on which it is called.

After a call to the fsetpos function on a stream that permits both reading and writing, the next file operation may be input or output.

Portability

ANSI

Returns

If successful, the fsetpos function returns 0. If it fails, the fsetpos function returns a nonzero value and stores an appropriate error code in the external integer errno .

Example

See the example for the fgetpos function.

See Also

fgetpos , fseek , ftell , lseek

1.156 fstat()

fstat-Get file status

Synopsis

```
  #include <sys/stat.h>

  rc = fstat(file, st);

  int rc;     /* return code  */
  int file;   /* UNIX file handle */
  struct stat *st;  /* stat info structure */
```

Description

This function obtains information for the given file handle. Permission to read, write, or execute the file is not required.

This function is provided for compatibility with UNIX.

It should only be called for files opened with the open function.

For code that will be used only on the Amiga, use the AmigaDOS function Examine instead.

The information is placed into the stat structure pointed to by the st argument. The stat structure is defined in the file stat.h.

st is a pointer to a stat structure that must be allocated on a 4-byte (long word) boundary by the calling program. A common error is failing to allocate the structure before calling the function. You can make sure the structure is long-word aligned by either declaring it with the __aligned keyword or by allocating it dynamically with any SAS/C or system allocation function (such as the malloc or AllocMem function).

The following table lists defines that are combined with the logical OR operator to form the st_mode field. This list is found in the file sys/commifmt.h.

Symbol	Meaning
S_ISCRIPT	The object has its script protection bit set.
S_IPURE	The object is an executable.
S_IARCHIVE	The file has its archive bit set.
S_IREAD	The file is readable.
S_IWRITE	The file is writable.
S_IEXECUTE	The file is executable.
S_IDELETE	The file is deletable.

Portability

UNIX

Returns

If the operation is successful, the function returns 0. Otherwise, it returns -1 and places error information in the external integers errno and _OSERR .

See Also

chmod , errno , _OSERR

1.157 ftell()

ftell-Get a level 2 file position

Synopsis

```
#include <stdio.h>

apos = ftell(fp);

long int apos;  /* absolute file position */
FILE *fp; /* file pointer */
```

```
Description

  The ftell function returns a long integer value that is the
  current byte position in the file, relative to the beginning.  It
  is equivalent to the following call:

     apos = lseek(fp->_file,0L,1);

  It is implemented as a function, not as a macro.

Portability

   ANSI

Returns

  For the ftell function, an error is indicated by a return value of
  -1L. The external integers  errno  and  _OSERR  contain additional
  error information.

See Also

   errno ,  fopen ,  lseek ,  _OSERR ,  tell
```

1.158 fwrite()

```
fwrite-Write blocks to a level 2 file

Synopsis

  #include <stdio.h>

  a = fwrite(b,bsize,n,fp);

  size_t a;       /* actual number of blocks */
  const void *b; /* pointer to first block */
  size_t bsize;  /* size of block in bytes */
  size_t n;       /* maximum number of blocks */
  FILE *fp;       /* file pointer */

Description

  This function performs level 2 I/O operations to write blocks of
  data.  Each block contains bsize bytes, and up to n blocks are
  stored into contiguous memory locations beginning at location b.

  Blocks are written until n blocks have been sent or until the
  output device cannot accept any more.  If the output device
  becomes full in the middle of a block, a partial block is written,
  but it is not included in the function return value.  In other
  words, the return value indicates the number of complete blocks
  that were written.

Portability
```

```
   ANSI
```

Returns

```
  This function returns the number of complete blocks that were
  processed.
```

See Also

```
   fclose ,  feof ,  ferror ,  fgetc ,  fopen ,  fputc ,  fread ,
   fseek
```

1.159 gcvt()

gcvt-Convert a floating-point number to a string

Synopsis

```
  #include <math.h>

  p = gcvt(v,dig,buffer);

  char *p;       /* points to buffer */
  double v;      /* floating point value */
  int dig;       /* number of significant digits */
  char *buffer;  /* output buffer */
```

Description

```
  This function converts the specified floating-point value into a
  null-terminated string in the output buffer.  The string will be
  in one of two formats.  First, the gcvt function attempts to
  produce dig significant digits in the FORTRAN F format.  If that
  fails, it produces dig significant digits in the FORTRAN E format.
  Trailing zeroes are eliminated, if necessary.

  Make sure that the specified buffer is large enough when using
  this function.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

```
   UNIX
```

Returns

```
  The function returns a pointer to the buffer.
```

Example

```
  /* This example displays 314150 */
```

```
#include <stdio.h>
#include <stdlib.h>

void main(void)
{
    char s[100];

    printf("gcvt(3.1415e5,7,s) = %s\n", gcvt(3.1415e5,7,s));
}
```

See Also

ecvt , fcvt

1.160 geta4()

geta4-Establish addressability to the global data area

Synopsis

```
#include <dos.h>

void geta4(void); range from -1 to 255.  The
function, however, will return a result for values above 255, but
the results are not necessarily correct and cannot be relied upon.

The reason -1 is included as a valid argument is to avoid a
nonsensical result if you feed the EOF value to one of this macro
or function.  EOF can be returned by the  getchar  function and
other I/O functions, and if you pass it to this function, the
return value will be 0.

When you include the ctype.h file, this function generates inline
code to test the static array named  __ctype .  This array contains
a bit mask for each of the 256 possible character values and for
the integer value -1.  If you do not include the ctype.h file, you
reduce your program size slightly at the expense of execution
speed.

The isascii function is not available if the  _STRICT_ANSI  flag has
been defined.
```

Portability

SAS/C

Returns

This function returns a nonzero value if the test is true and 0 if the test is false.

See Also

__ctype

1.161 iscntrl()

```
iscntrl-Test if control character

Synopsis

  #include <ctype.h>

  t = iscntrl(c);   /* Test if control character               */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */

Description

  This function tests for control characters.  If you include
  the file ctype.h, the functions are defined as macros. If you do
  not include the file ctype.h, this function is resolved in the
  library.  If you want to use the function version (not the
  macro) but must include the file ctype.h for some other reason,
  use an #undef statement to undefine this macro

      #undef iscntrl

  You can use either characters or integers as arguments, but this
  macro is defined only over the integer range from -1 to 255.  The
  function, however, will return a result for values above 255, but
  the results are not necessarily correct and cannot be relied upon.

  The reason -1 is included as a valid argument is to avoid a
  nonsensical result if you feed the EOF value to one of this macro
  or function.  EOF can be returned by the  getchar  function and
  other I/O functions, and if you pass it to this function, the
  return value will be 0.

  When you include the ctype.h file, this function generates inline
  code to test the static array named  __ctype .  This array contains
  a bit mask for each of the 256 possible character values and for
  the integer value -1.  If you do not include the ctype.h file, you
  reduce your program size slightly at the expense of execution
  speed.

Portability

   ANSI

Returns

  This function returns a nonzero value if the test is true and 0 if
  the test is false.

See Also

   __ctype
```

1.162 iscsym()

```
iscsym-Test if C symbol character

Synopsis

  #include <ctype.h>

  t = iscsym(c);    /* Test if C symbol character              */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */

Description

  This function tests for C symbol characters.  If you include the
  file ctype.h, the functions are defined as macros. If you do not
  include the file ctype.h, this function is resolved in the
  library.  If you want to use the function version (not the macro)
  but must include the file ctype.h for some other reason, use an
  #undef statement to undefine this macro

      #undef iscsym

  You can use either characters or integers as arguments, but this
  macro is defined only over the integer range from -1 to 255.  The
  function, however, will return a result for values above 255, but
  the results are not necessarily correct and cannot be relied upon.

  The reason -1 is included as a valid argument is to avoid a
  nonsensical result if you feed the EOF value to one of this macro
  or function.  EOF can be returned by the  getchar  function and
  other I/O functions, and if you pass it to this function, the
  return value will be 0.

  When you include the ctype.h file, this function generates inline
  code to test the static array named  __ctype .  This array contains
  a bit mask for each of the 256 possible character values and for
  the integer value -1.  If you do not include the ctype.h file, you
  reduce your program size slightly at the expense of execution
  speed.

  The iscsym function is not available if the  _STRICT_ANSI  flag has
  been defined.

Portability

   SAS/C

Returns

  This function returns a nonzero value if the test is true and 0 if
  the test is false.

See Also
```

```
   __ctype
```

1.163 iscsymf()

```
iscsymf-Test if C symbol lead character

Synopsis

  #include <ctype.h>

  t = iscsymf(c);   /* Test if C symbol lead character          */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */

Description

  This function tests for C leading symbol characters.  If you
  include the file ctype.h, the functions are defined as macros.  If
  you do not include the file ctype.h, this function is resolved in
  the library.  If you want to use the function version (not the
  macro) but must include the file ctype.h for some other reason,
  use an #undef statement to undefine this macro

      #undef iscsymf

  You can use either characters or integers as arguments, but this
  macro is defined only over the integer range from -1 to 255.  The
  function, however, will return a result for values above 255, but
  the results are not necessarily correct and cannot be relied upon.

  The reason -1 is included as a valid argument is to avoid a
  nonsensical result if you feed the EOF value to one of this macro
  or function.  EOF can be returned by the  getchar  function and
  other I/O functions, and if you pass it to this function, the
  return value will be 0.

  When you include the ctype.h file, this function generates inline
  code to test the static array named  __ctype .  This array contains
  a bit mask for each of the 256 possible character values and for
  the integer value -1.  If you do not include the ctype.h file, you
  reduce your program size slightly at the expense of execution
  speed.

  The iscsymf function is not available if the  _STRICT_ANSI  flag has
  been defined.

Portability

   SAS/C

Returns

  This function returns a nonzero value if the test is true and 0 if
  the test is false.
```

See Also

```
   __ctype
```

1.164 isdigit()

isdigit-Test if decimal digit character (0 to 9)

Synopsis

```
  #include <ctype.h>

  t = isdigit(c);   /* Test if decimal digit character (0 to 9) */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */
```

Description

```
  This function tests for digit characters.  If you include the file
  ctype.h, the functions are defined as macros.  If you do not
  include the file ctype.h, this function is resolved in the
  library.  If you want to use the function version (not the macro)
  but must include the file ctype.h for some other reason, use an
  #undef statement to undefine this macro

      #undef isdigit

  You can use either characters or integers as arguments, but this
  macro is defined only over the integer range from -1 to 255.  The
  function, however, will return a result for values above 255, but
  the results are not necessarily correct and cannot be relied upon.

  The reason -1 is included as a valid argument is to avoid a
  nonsensical result if you feed the EOF value to one of this macro
  or function.  EOF can be returned by the  getchar  function and
  other I/O functions, and if you pass it to this function, the
  return value will be 0.

  When you include the ctype.h file, this function generates inline
  code to test the static array named  __ctype .  This array contains
  a bit mask for each of the 256 possible character values and for
  the integer value -1.  If you do not include the ctype.h file, you
  reduce your program size slightly at the expense of execution
  speed.
```

Portability

```
   ANSI
```

Returns

```
  This function returns a nonzero value if the test is true and 0 if
  the test is false.
```

```
See Also

   __ctype
```

1.165 isgraph()

```
isgraph-Test if graphic character

Synopsis

  #include <ctype.h>

  t = isgraph(c);   /* Test if graphic character                */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */

Description

  This function tests for graphic characters.  If you include the
  file ctype.h, the functions are defined as macros.  If you do not
  include the file ctype.h, this function is resolved in the
  library.  If you want to use the function version (not the macro)
  but must include the file ctype.h for some other reason, use an
  #undef statement to undefine this macro

      #undef isgraph

  You can use either characters or integers as arguments, but this
  macro is defined only over the integer range from -1 to 255.  The
  function, however, will return a result for values above 255, but
  the results are not necessarily correct and cannot be relied upon.

  The reason -1 is included as a valid argument is to avoid a
  nonsensical result if you feed the EOF value to one of this macro
  or function.  EOF can be returned by the  getchar  function and
  other I/O functions, and if you pass it to this function, the
  return value will be 0.

  When you include the ctype.h file, this function generates inline
  code to test the static array named  __ctype .  This array contains
  a bit mask for each of the 256 possible character values and for
  the integer value -1.  If you do not include the ctype.h file, you
  reduce your program size slightly at the expense of execution
  speed.

Portability

   ANSI

Returns

  This function returns a nonzero value if the test is true and 0 if
  the test is false.
```

See Also

   __ctype

1.166 islower()

islower-Test if lower case character

Synopsis

```
  #include <ctype.h>

  t = islower(c);   /* Test if lower case character             */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */
```

Description

  This function tests for lowercase characters. If you include the file ctype.h, the functions are defined as macros. If you do not include the file ctype.h, this function is resolved in the library. If you want to use the function version (not the macro) but must include the file ctype.h for some other reason, use an #undef statement to undefine this macro

```
      #undef islower
```

  You can use either characters or integers as arguments, but this macro is defined only over the integer range from -1 to 255. The function, however, will return a result for values above 255, but the results are not necessarily correct and cannot be relied upon.

  The reason -1 is included as a valid argument is to avoid a nonsensical result if you feed the EOF value to one of this macro or function. EOF can be returned by the getchar function and other I/O functions, and if you pass it to this function, the return value will be 0.

  When you include the ctype.h file, this function generates inline code to test the static array named __ctype . This array contains a bit mask for each of the 256 possible character values and for the integer value -1. If you do not include the ctype.h file, you reduce your program size slightly at the expense of execution speed.

Portability

   ANSI

Returns

  This function returns a nonzero value if the test is true and 0 if the test is false.

See Also

__ctype

1.167 isprint()

isprint-Test if printable character

Synopsis

```
  #include <ctype.h>

  t = isprint(c);   /* Test if printable character              */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */
```

Description

This function tests for printable characters. If you include the file ctype.h, the functions are defined as macros. If you do not include the file ctype.h, this function is resolved in the library. If you want to use the function version (not the macro) but must include the file ctype.h for some other reason, use an #undef statement to undefine this macro

```
    #undef isprint
```

You can use either characters or integers as arguments, but this macro is defined only over the integer range from -1 to 255. The function, however, will return a result for values above 255, but the results are not necessarily correct and cannot be relied upon.

The reason -1 is included as a valid argument is to avoid a nonsensical result if you feed the EOF value to one of this macro or function. EOF can be returned by the getchar function and other I/O functions, and if you pass it to this function, the return value will be 0.

When you include the ctype.h file, this function generates inline code to test the static array named __ctype . This array contains a bit mask for each of the 256 possible character values and for the integer value -1. If you do not include the ctype.h file, you reduce your program size slightly at the expense of execution speed.

Portability

ANSI

Returns

This function returns a nonzero value if the test is true and 0 if the test is false.

See Also

```
   __ctype
```

1.168 ispunct()

ispunct-Test if punctuation character

Synopsis

```
  #include <ctype.h>

  t = ispunct(c);   /* Test if punctuation character            */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */
```

Description

This function tests for punctuation characters. If you include the file ctype.h, the functions are defined as macros. If you do not include the file ctype.h, this function is resolved in the library. If you want to use the function version (not the macro) but must include the file ctype.h for some other reason, use an #undef statement to undefine this macro

```
      #undef ispunct
```

You can use either characters or integers as arguments, but this macro is defined only over the integer range from -1 to 255. The function, however, will return a result for values above 255, but the results are not necessarily correct and cannot be relied upon.

The reason -1 is included as a valid argument is to avoid a nonsensical result if you feed the EOF value to one of this macro or function. EOF can be returned by the getchar function and other I/O functions, and if you pass it to this function, the return value will be 0.

When you include the ctype.h file, this function generates inline code to test the static array named __ctype . This array contains a bit mask for each of the 256 possible character values and for the integer value -1. If you do not include the ctype.h file, you reduce your program size slightly at the expense of execution speed.

Portability

ANSI

Returns

This function returns a nonzero value if the test is true and 0 if the test is false.

See Also

```
   __ctype
```

1.169 isspace()

isspace-Test if space character

Synopsis

```
  #include <ctype.h>

  t = isspace(c);   /* Test if space character                 */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */
```

Description

This function tests for space characters. If you include the file ctype.h, the functions are defined as macros. If you do not include the file ctype.h, this function is resolved in the library. If you want to use the function version (not the macro) but must include the file ctype.h for some other reason, use an #undef statement to undefine this macro

```
      #undef isspace
```

You can use either characters or integers as arguments, but this macro is defined only over the integer range from -1 to 255. The function, however, will return a result for values above 255, but the results are not necessarily correct and cannot be relied upon.

The reason -1 is included as a valid argument is to avoid a nonsensical result if you feed the EOF value to one of this macro or function. EOF can be returned by the getchar function and other I/O functions, and if you pass it to this function, the return value will be 0.

When you include the ctype.h file, this function generates inline code to test the static array named __ctype . This array contains a bit mask for each of the 256 possible character values and for the integer value -1. If you do not include the ctype.h file, you reduce your program size slightly at the expense of execution speed.

Portability

ANSI

Returns

This function returns a nonzero value if the test is true and 0 if the test is false.

See Also

  __ctype

1.170 isupper()

isupper-Test if upper case character

Synopsis

```
  #include <ctype.h>

  t = isupper(c);   /* Test if upper case character             */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */
```

Description

This function tests for UPPERCASE characters. If you include the file ctype.h, the functions are defined as macros. If you do not include the file ctype.h, this function is resolved in the library. If you want to use the function version (not the macro) but must include the file ctype.h for some other reason, use an #undef statement to undefine this macro

```
      #undef isupper
```

You can use either characters or integers as arguments, but this macro is defined only over the integer range from -1 to 255. The function, however, will return a result for values above 255, but the results are not necessarily correct and cannot be relied upon.

The reason -1 is included as a valid argument is to avoid a nonsensical result if you feed the EOF value to one of this macro or function. EOF can be returned by the getchar function and other I/O functions, and if you pass it to this function, the return value will be 0.

When you include the ctype.h file, this function generates inline code to test the static array named __ctype . This array contains a bit mask for each of the 256 possible character values and for the integer value -1. If you do not include the ctype.h file, you reduce your program size slightly at the expense of execution speed.

Portability

  ANSI

Returns

This function returns a nonzero value if the test is true and 0 if the test is false.

See Also

   __ctype

1.171 isxdigit()

isxdigit-Test if hex digit character

Synopsis

```
  #include <ctype.h>

  t = isxdigit(c);  /* Test if hex digit character              */
        /*  (0 to 9, A to F, a to f)              */

  int t;   /* 0 if false, non-zero if true */
  int c:   /* character to test */
```

Description

  This function tests for hexidecimal digit characters. If you
  include the file ctype.h, the functions are defined as macros. If
  you do not include the file ctype.h, this function is resolved in
  the library. If you want to use the function version (not the
  macro) but must include the file ctype.h for some other reason,
  use an #undef statement to undefine this macro

```
      #undef isxdigit
```

  You can use either characters or integers as arguments, but this
  macro is defined only over the integer range from -1 to 255. The
  function, however, will return a result for values above 255, but
  the results are not necessarily correct and cannot be relied upon.

  The reason -1 is included as a valid argument is to avoid a
  nonsensical result if you feed the EOF value to one of this macro
  or function. EOF can be returned by the getchar function and
  other I/O functions, and if you pass it to this function, the
  return value will be 0.

  When you include the ctype.h file, this function generates inline
  code to test the static array named __ctype . This array contains
  a bit mask for each of the 256 possible character values and for
  the integer value -1. If you do not include the ctype.h file, you
  reduce your program size slightly at the expense of execution
  speed.

Portability

   ANSI

Returns

  This function returns a nonzero value if the test is true and 0

```
  if the test is false.

See Also

   __ctype
```

1.172 isatty()

```
isatty-Test a file descriptor for a terminal device

Synopsis

  #include <fcntl.h>

  rc = isatty(fd)

  int fd;   /*  level 1 file descriptor */
  int rc;   /*  return code */

Description

  This function takes a file descriptor as returned from a call to
  the level 1 file I/O function  open  and tests to see if the file
  descriptor is associated with a terminal device (such as a console
  window).

Portability

   UNIX

Returns

  If the file descriptor is associated with a terminal device, the
  routine returns 1.  Otherwise, it returns 0.

See Also

   open
```

1.173 jrand48()

```
jrand48-Generate a random long integer (external seed)

Synopsis

  #include <math.h>

  z = jrand48(seed);

  long z;     /* random long */
  unsigned short seed[3]; /* seed value (high bits in seed[0]) */
```

```
Description

  This function generates random numbers using the linear
  congruential algorithm and 48-bit arithmetic. The jrand48 function
  is provided for cases where several seeds are in use at the same
  time, so you can specify the seed on each function call.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   UNIX

Returns

  The jrand48 function returns signed long integers uniformly
  distributed over the interval from -2**31 to 2**31-1.

See Also

   lcong48 ,  mrand48 ,  rand ,  srand ,  srand48
```

1.174 labs()

```
labs-Long integer absolute value

Synopsis

  #include <stdlib.h>

  al = labs(l);

  long int al,l;

Description

  This macro computes the absolute value of a long integer. It
  generates inline code to perform the conversion.  The definition
  is

     #define labs(i)  ((i)<0?-(i):(i))

Portability

   ANSI

Returns

  This function returns a long integer holding the absolute value of
  the parameter.

See Also
```

abs , fabs , iabs

1.175 lcong48()

lcong48-Set linear congruence parameters

Synopsis

```
  #include <math.h>

  void lcong48(parm);

  unsigned short parm[7]; /* parameters */
```

Description

The lcong48 function allows an intricate initialization of the linear congruential algorithm.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

UNIX

See Also

jrand48 , rand , seed48 , srand , srand48

1.176 ldexp()

ldexp-Combine floating-point value

Synopsis

```
  #include <math.h>

  v = ldexp(d,x);

  double v; /* value */
  double d; /* fraction */
  int x;    /* exponent */
```

Description

The ldexp function adds the integer x to the exponent in the argument d, which is the same as computing:

```
     v = d * (2 ** x)
```

If d and x are the results of the frexp function, then the ldexp

  function performs the reverse of the frexp function. Also, if the
  absolute value of the resulting exponent is greater than 1,023,
  then the __matherr function is called with an overflow or
  underflow error indication.

Portability

   ANSI

Returns

  This function returns a double-precision floating-point number
  holding the result of the above equation.

See Also

   fmod , frexp , __matherr , modf

1.177 ldiv()

ldiv-Return the long integer quotient and the remainder

Synopsis

```
  #include <stdlib.h>

  res = ldiv(numer, denom);

  ldiv_t res;    /* resulting quotient and remainder */
  long int numer;    /* numerator for the divide         */
  long int denom;    /* denominator for the divide       */
```

Description

  This function returns the quotient and the remainder obtained from
  performing a divide on long integers.

Portability

   ANSI

Returns

  This function returns a structure containing both the quotient and
  the remainder.

  The structure is defined in the stdlib.h file as follows:

```
      typedef struct
      {
    long int quot;
    long int rem;
      } ldiv_t;
```

Example

```
/*
 * Obtain both quotient and remainder
 * for a value divided by 10
 */
#include <stdio.h>
#include <stdlib.h>

void quotrem(long val)
{
    ldiv_t result;

    result = ldiv(val, 10L);

    printf("Quotient  = %ld\n", result.quot);
    printf("Remainder = %ld\n", result.rem);
}

void main(void)
{
    quotrem(42);
}
```

See Also

 div

1.178 localeconv()

localeconv-Return information on locale formatting conventions

Synopsis

```
#include <locale.h>

lcl = localeconv(void);

struct lconv *lcl;     /* Locale information structure */
```

Description

This function fills in a structure of information about numeric
and monetary formatting for the current program's locale. The
structure is defined in the file locale.h as follows:

```
  struct lconv {
     char *decimal_point;
     char *thousands_sep;
     char *grouping;

   #define LCONVM int_curr_symbol
     char *int_curr_symbol;   /* international currency symbol  */
            /* for current locale   */
     char *currency_symbol;   /* local currency symbol for  */
            /* current locale     */
```

```
    char *mon_decimal_point; /* decimal point for monetary */
           /* quantities     */
    char *mon_thousands_sep; /* separator for groups of digits */
           /* in monetary quantities   */
    char *mon_grouping;      /* size of digit groups in  */
           /* monetary quantities    */
    char *positive_sign;     /* string indicating non-   */
           /* negative monetary quantity */
    char *negative_sign;     /* string indicating negative */
           /* monetary quantity    */
    char int_frac_digits;    /* number of digits after decimal */
           /* point in international   */
           /* monetary quantities    */
    char frac_digits;        /* number of digits after decimal */
           /* point in monetary quantities */
    char p_cs_precedes;      /* 1=currency symbol precedes */
           /* nonnegative monetary quantity  */
           /* 0=symbol succeeds quantity */
    char p_sep_by_space;     /* 1=space between currency symbol*/
           /* and non-negative monetary  */
           /* quantity     */
           /* 0=no space     */
    char n_cs_precedes;      /* 1=currency symbol precedes */
           /* negative monetary quantity */
           /* 0=symbol succeeds quantity */
    char n_sep_by_space;     /* 1=space between currency symbol*/
           /* and negative monetary quantity */
           /* 0=no space     */
    char p_sign_posn;        /* position of sign for positive  */
           /* monetary quantities    */
    char n_sign_posn;        /* position of sign for negative  */
           /* monetary quantities    */
 };
```

The decimal point character used to format nonmonetary values defaults to a period (.). The character used to separate groups of digits before the decimal point character in formatted nonmonetary values defaults to a comma (,).

Portability

ANSI

Returns

This function returns a pointer to the lconv structure for the current locale.

See Also

setlocale

1.179 localtime()

```
localtime-Unpack local time

Synopsis

  #include <time.h>

  ut = localtime(t);

  struct tm *ut;
  const time_t *t;

Description

  This function unpacks a time value from the long integer form into
  a structure.  Normally, the time value represents the number of
  seconds since 00:00:00, January 1, 1970, Greenwich Mean Time. (The
   time  function returns this kind of number.) The localtime function
  adjusts the number for the local time zone.

  This function expects a pointer as the argument. A common error is
  to pass the actual time value instead of the pointer.

   The functions localtime,  gmtime ,  ctime , and  mktime  share a
   static data area.  A call to any one of these destroys the
   results of the previous call.

  The  tm structure  is defined in the file time.h.

Portability

   ANSI

Example

  #include <stdio.h>
  #include <time.h>

  void main(void)
  {
      struct tm *p;
      long t;

      time(&t);
      p = localtime(&t);
      printf("Local time is %s\n",asctime(p));
  }

See Also

   asctime ,  ctime ,  gmtime ,  time
```

1.180 log()

```
log-Natural logarithm function
```

Synopsis

```
  #include <math.h>

  r = log(x);

  double r, x;
```

Description

```
  The log function calculates the base E logarithm.  This function
  requires a positive argument.  If you enter a negative argument,
  the  __matherr  function is called with a DOMAIN error.
```

Portability

```
   ANSI
```

Returns

```
  This function returns a double-precision floating-point number
  that contains the base E logarithm of the parameter.
```

See Also

```
   log10 ,  __matherr
```

1.181 log10()

log10-Base 10 logarithm function

Synopsis

```
  #include <math.h>

  r = log10(x);

  double r, x;
```

Description

```
  The log10 function calculates the base 10 logarithm.  This
  function requires a positive argument.  If you enter a negative
  argument, the  __matherr  function is called with a DOMAIN error.
```

Portability

```
   ANSI
```

Returns

```
  This function returns a double-precision floating-point number
  that is the base 10 logarithm of the parameter.
```

See Also

```
   log ,  __matherr
```

1.182 longjmp()

longjmp-Perform a long jump

Synopsis

```
  #include <setjmp.h>

  void longjmp(save,value);

  jmp_buf save; /* address of save area */
  int value;  /* return value */
```

Description

  The setjmp function saves the current stack mark in a specified
  save area and returns a code of 0. A subsequent call to the
  longjmp function with the same save area will then cause control
  to return to the next statement after the original setjmp call,
  with value as the return code. If the return code is 0, it is
  forced to 1 by the longjmp function.

  This mechanism is useful for quickly popping back up through
  multiple layers of function calls under exceptional circumstances.

  Do not call the longjmp function with an invalid save area. It
  may disrupt the system. Do not use the longjmp function after the
  function calling the setjmp function has returned to its caller
  because the stack frame for that function no longer exists.

  From within a shared library, you must not call any library
  functions that terminate your program. For example, you cannot
  call exit , __exit , or abort from a shared library. You also
  cannot use setjmp and longjmp to jump across a call from the
  program into the library.

Portability

   ANSI

See Also

```
   exit ,  setjmp
```

1.183 lqsort()

lqsort-Sort an array of long integers

```
Synopsis

  #include <stdlib.h>

  void lqsort(la,n);

  long *la;   /* pointer to long int array */
  size_t n;   /* number of elements in array */

Description

  The lqsort function sorts the specified array of long integers
  using the ACM 271 algorithm, more popularly known as Quicksort.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

See Also

   dqsort ,  fqsort ,  qsort ,  sqsort ,  tqsort
```

1.184 lrand48()

```
lrand48-Generate a random positive long integer (internal seed)

Synopsis

  #include <math.h>

  y = lrand48(void);

  long y;        /* random positive long */

Description

  This function generates random numbers using the linear
  congruential algorithm and 48-bit arithmetic.  The lrand48
  function uses an internal 48-bit storage area for the seed value.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   UNIX

Returns

  The lrand48 function returns nonnegative long integers uniformly
  distributed over the interval from -2**31 to 2**31-1.
```

See Also

```
   nrand48 ,  rand ,  srand ,  srand48
```

1.185 lsbrk()

lsbrk-Allocate memory

Synopsis

```
  #include <stdlib.h>

  p = lsbrk(lbytes);

  void *p;  /* block pointer */
  long lbytes;  /* number of bytes */
```

Description

The lsbrk function requests lbytes of memory from the system, adding the allocated block to a linked list of memory blocks to be returned to the system when the program terminates.

This function is provided for compatibility with previous versions of the compiler.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

OLD

Returns

An error is indicated by a NULL pointer. The lsbrk function returns the address of the block just allocated.

See Also

```
   getmem ,  malloc ,  rbrk ,  sbrk
```

1.186 lseek()

lseek-Set a level 1 file position

Synopsis

```
  #include <fcntl.h>

  apos = lseek(fh,rpos,mode);
  long apos;  /* absolute file position */
  int fh;     /* file handle */
  long rpos;  /* relative file position */
  int mode;   /* seek mode */
```

Description

This function moves the byte cursor of a level 1 file to a new position. The mode argument must be one of the following:

0
seek from the beginning of the file. The rpos argument is the number of bytes from the beginning of the file. This value must be positive.

1
seek from the current position of the file. The rpos argument is the number of bytes relative to the current position. This value can be positive or negative.

2
seek from the end of the file. The rpos argument is the number of bytes relative to the end of the file. This value must be negative or 0.

If the lseek function is asked to move 0 bytes relative to the current position, it simply returns the current file position.

Returns

This function returns -1 if an error occurs, in which case the external integers errno and _OSERR contain additional error information.

Portability

UNIX

Example

```
  /**
   * This program totals the number of bytes used by
   * all normal files in the current directory.
   **/

  #include <stdio.h>
  #include <stdlib.h>
  #include <fcntl.h>   /* for Level 1 I/O */

  char names[8192];    /* holds file names */

  void main(void)
  {
      char *p;
      int f,n;
      long x,y;
      if (getfnl("#?",names,sizeof(names),0) <= 0)
      {
    printf("Can’t build file name list\n");
    exit(EXIT_FAILURE);
      }
      for (x=0, n=0, p=names; *p!=’\0’; p+=strlen(p)+1)
      {
    f = open(p,O_RDONLY);
    if (f < 0)
    {
        printf("Can’t open \"%s\"\n",p);
        exit(EXIT_FAILURE);
    }
    y = lseek(f,0L,2);
    if (y < 0)
    {
        printf("Seek failure on \"%s\"\n",p);
        exit(EXIT_FAILURE);
    }
    x += y;
    n++;
    close(f);
      }
      printf("%d files, %ld bytes used\n",n,x);
  }
```

See Also

   errno , open , _OSERR , tell

1.187 lstat()

lstat-Get file status

Synopsis

```
  #include <sys/stat.h>

  rc = lstat(file, st);

  int rc;     /* return code  */
  const char *file; /* file name */
  struct stat *st;  /* stat info structure */
```

Description

  This function obtains information for the given file. If the file
  is not in the current directory, the file path must be included as
  part of the filename. Permission to read, write, or execute the
  file is not required.

  If the file referred to is a link, this function returns
  information on the link instead of the file to which it is linked.

  This function works under all revisions of the operating system and is provided for compatibility with UNIX. For code that will be used only on the Amiga, use the AmigaDOS function Examine instead.

The information is placed into the stat structure pointed to by the st argument. The structure is defined in the file stat.h.

st is a pointer to a stat structure that must be allocated on a 4-byte (long word) boundary by the calling program. A common error is failing to allocate the structure before calling the function. You can make sure the structure is long-word aligned by either declaring it with the __aligned keyword or by allocating it dynamically with any SAS/C or system allocation function (such as the malloc or AllocMem function).

The following table lists defines that are combined with the logical OR operator to form the st_mode field. This list is found in the file sys/commifmt.h.

Symbol	Meaning
S_ISCRIPT	The object has its script protection bit set.
S_IPURE	The object is an executable.
S_IARCHIVE	The file has its archive bit set.
S_IREAD	The file is readable.
S_IWRITE	The file is writable.
S_IEXECUTE	The file is executable.
S_IDELETE	The file is deletable.

Portability

UNIX

Returns

If the operation is successful, the function returns 0. Otherwise, it returns -1 and places error information in the external integers errno and _OSERR .

See Also

chmod , errno , _OSERR

1.188 __main()

__main-Standard preprocessing for the main module

Synopsis

```
#include <stdlib.h>

void __stdargs __main(line);

char *line;   /* ptr to command line that caused execution */
```

Description

```
  The __main function performs the standard preprocessing for the
  main module of a C program.  It accepts a command line of the
  following form:

      program-name arg1 arg2

  It builds a list of pointers to each argument and the first
  pointer is to the program name.  The __main function also opens
  the standard I/O files stdin, stdout, and stderr. __main calls the
  function main with the standard argc and argv parameters.

  Unlike previous editions of the compiler, this function is
  declared with the __stdargs keyword.

  The source code for this function is in the file _main.c in the
  sc:source directory.

  For more information on __main, refer to Chapter 10, "Using
  Startup Modules, Autoinitialization, and Autotermination
  Functions," in SAS/C Development System User's Guide, Volume 1:
  Introduction, Editor, Compiler.
```

Portability

```
   SAS/C
```

See Also

```
   main
```

1.189 main()

main-Your main or principal function

Synopsis

```
  #include <workbench/startup.h>

  int main(argc,argv);

  int argc;        /* argument count */
  char *argv[];
```

Description

```
  This function does not actually exist in the library; you must
  supply one of these main programs in each of your applications.
  If you trace through the two startup modules c.a and _main.c, you
  will find that the module c.a passes control to the module
  _main.c, which then calls the function named main. Since the
  source code for both of these modules is supplied, you are free to
  change this initialization procedure for special applications.
```

The standard version simulates UNIX's interface with C programs by setting up a vector, which is simply an array of pointers.

The argv array contains pointers to the command-line arguments, and the argument argc indicates how many pointers are in the array. For example, you can start the program myprog with the following command:

```
myprog abc def "ghi jkl"
```

Then, the startup code sets up the argv array as follows:

```
argv[0] => "myprog"
argv[1] => "abc"
argv[2] => "def"
argv[3] => "ghi jkl"
argv[4] => NULL
```

The argc argument contains the value 4.

Under Workbench, there is no command line. In this case, the argument argc is 0 indicating no command or arguments, and the argument argv is actually a pointer to the Workbench startup message structure. You can convert it with a simple cast:

```
#include <workbench/startup.h>

struct WBStartup *Wbs;

Wbs = (struct WBStartup *)argv;
```

Portability

ANSI

Returns

When the main function returns to its caller (normally the _main.c function), the program exits to AmigaDOS with a termination code of the value returned by main.

If you want to pass a nonzero termination code back to AmigaDOS, use the exit or __exit function, or return a nonzero return code from your main function.

Example

```
/* This program is intended to run only under the    */
/* Shell and displays the command and any arguments  */

#include <stdio.h>

int main(int argc, char *argv[])
{
    int i;

    printf("command = %s\n",argv[0]);
    for (i = 0; argc > 0; i++, argc--)
  printf("argument %d = %s\n",i,argv[i]);
    return(0);
}


/* This program is intended to run only under WorkBench and */
/* gets its arguments from the WorkBench message structure  */

#include <stdlib.h>
#include <stdio.h>
#include <workbench/startup.h>

int main(int argc, char *argv[])
{
    struct WBStartup *wbs;
    int i;

    if (argc != 0)
  exit(EXIT_FAILURE);

    wbs = (struct WBStartup *)argv;

    printf("command = %s\n", wbs->sm_ArgList[0].wa_Name);
    for (i = 1; i < wbs->sm_NumArgs; i++)
  printf("argument %d = %s\n", i, wbs->sm_ArgList[i].wa_Name);
    return(0);
}


/*  This program runs correctly under either Workbench or */
/*  the Shell and can be used with stack or registerized  */
/*  parameters.             */

#include <stdio.h>
#include <workbench/startup.h>

int main (int argc, char *argv[])
{
    struct WBStartup *msg;
    int i;

    if (argc != 0)
    {
  printf("command = %s\n", argv[0]);
  for (i=0; argc > 0; i++, argc--)
      printf("argument %d = %s\n", i, argv[i]);
    }
    else
    {
  msg = (struct WBStartup *)argv;
  printf("command = %s\n", msg->sm_ArgList[0].wa_Name);
  for (i=1; i < msg->sm_NumArgs; i++)
      printf("argument %d = %s\n", i,
       msg->sm_ArgList[i].wa_Name);
    }
    return(0);
  }
```

See Also

```
   exit ,  __exit ,  __main
```

1.190 malloc()

malloc-Allocate memory

Synopsis

```
  #include <stdlib.h>

  b = malloc(n);

  void *b;      /*block pointer   */
  size_t n;     /*number of bytes */
```

Description

The malloc function allocates a block that is n bytes long and is aligned in such a way that you can cast the block pointer to any pointer type. If the block cannot be allocated, a NULL pointer is returned.

The malloc function can only allocate 64 kilobytes at a time if short integers are used.

Portability

ANSI

Returns

A NULL pointer is returned if there is not enough space for the requested block.

Example

```
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>

  struct LIST
  {
      struct LIST *next;
      char text[2];
  };

  void main(int argc, char *argv[])
  {
      struct LIST *p;
      struct LIST *q;
      struct LIST list;
      char b[256];
      int x;

      printf("\nBegin new group...\n");
      for (q = &list; ; q = p)
      {
    printf("Enter a text string: ");
    if (fgets(b,sizeof(b),stdin) == NULL)
    {
        break;
    }
    if (b[0] == NULL)
    {
        if (q == &list)
        {
      printf("\n");
      exit(EXIT_SUCCESS);
        }
        break;
    }
    x = sizeof(struct LIST) - 2 + strlen(b) + 1;
    p = malloc(x);
    if (p == NULL)
    {
        printf("No more memory\n");
        exit(EXIT_FAILURE);
    }
    q->next = p;
    p->next = NULL;
    strcpy(p->text, b);
      }
      printf("\n\nTEXT LIST...\n");
      p = list.next;
      while(p != NULL)
      {
    q = p->next;
    printf("%s", p->text);
    free(p);
    p = q;
      }
      list.next = NULL;
  }
```

See Also

calloc , free , getmem , rbrk , realloc , rlsmem , sbrk

1.191 __matherr()

__matherr-Math error handler

Synopsis

```
  #include <math.h>
  a = __matherr(x);

  int a;        /* action code      */
  struct __exception *x;    /* exception vector */
```

Description

  The __matherr function is called whenever one of the higher-level
  math functions detects an error. The exception vector structure
  is defined in the file math.h and contains information about the
  error as follows:

```
     struct __exception
     {
        int type;       /* error type      */
        char *name;     /* math function name    */
        double arg1, arg2;    /* function arguments    */
        double retval;      /* proposed return value */
     };
```

  The codes for the type field in struct __exception are in the file
  math.h.

  The standard library version of the __matherr function translates
  the error type into a UNIX error code that is placed into the
  external integer errno . Then the function returns an action code
  of 0 to indicate that the math function should simply use the
  proposed return value. In other words, the math function will
  pass that value back to its caller.

  The SAS/C Compiler software includes source code in the source
  directory for the __matherr function, so you can change it to do
  more sophisticated error correction. One typical change is to
  place a different return value into the exception vector and then
  return a nonzero action code. This informs the math function that
  the return value has been changed. You must compile the modified
  matherr.c file and link the resultant object module with your code
  to incorporate the changes.

Portability

   UNIX

Returns

  For the __matherr function, a nonzero return indicates that the
  proposed return value in the exception vector has been changed and
  that the new value should be used. A return of 0 indicates that
  the proposed return value is acceptable.

See Also

  __except, _FPERR

1.192 max()

```
max-Compute the maximum of two values

Synopsis

  #include <math.h>

  v = max(a,b);

Description

  This macro computes the maximum of two arithmetic values.  It is
  defined as follows:

      #define max(a,b) ((a)>(b)?(a):(b))

  The max macro works with any arithmetic type or combination of
  types.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   UNIX

Returns

  This macro returns the larger of the two parameters.

See Also

   min
```

1.193 mblen()

```
mblen-Determine the length of a multibyte character

Synopsis

  #include <stdlib.h>

  length = mblen(s, n);

  int length;    /* length or state information */
  const char *s; /* pointer to characters or NULL */
  size_t  n;     /* maximum number of characters to look at */

Description

  This function determines the number of bytes comprising the
  multibyte character pointed to by the argument s, and it is useful
  for determining how much storage to allocate before calling the
```

mbstowcs function.

Portability

ANSI

Returns

If a NULL pointer is passed for the argument s, the return value indicates whether the current locale has state-dependent encodings. A nonzero value indicates that it does.

In the Amiga implementation, for all other values for the argument s, a 1 is returned since the implementation does not support multibyte characters.

See Also

mbstowcs , mbtowc

1.194 mbstowcs()

mbstowcs-Convert a multibyte string to a wide character string

Synopsis

```
#include <stdlib.h>

length = mbstowcs(pwcs, s, n);

size_t length;  /* length or state information       */
wchar_t *pwcs;  /* pointer to wide character string    */
const char *s;  /* pointer to characters or NULL     */
size_t n; /* maximum number of characters to look at */
```

Description

This function converts a multibyte string to a wide character string.

The mbstowcs function for the current locale is passed the input parameters and the result is returned.

Portability

ANSI

Returns

This function returns the length of the result string.

See Also

mblen , mbtowc

1.195 mbtowc()

```
mbtowc-Map a multibyte character to a wide character

Synopsis

  #include <stdlib.h>

  length = mbtowc(pwc, s, n);

  int length; /* length or state information       */
  wchar_t *pwc; /* pointer to wide character       */
  const char *s;  /* pointer to characters or NULL     */
  size_t n; /* maximum number of characters to look at */

Description

  This function maps a multibyte character to a wide character.  The
  output buffer must be long enough to hold the result. You can
  determine the length by calling the  mblen  function.

Portability

   ANSI

Returns

  This function returns the length of the multibyte character
  defined by the locale information.  If the argument s is NULL or
  if the argument n is equal to 0, this function returns 0.

See Also

   mblen ,  mbstowcs
```

1.196 memccpy()

```
memccpy-Copy a memory block

Synopsis

  #include <string.h>

  s = memccpy(to,from,c,n);

  void *s;    /* return pointer */
  void *to;   /* destination pointer */
  const void *from; /* source pointer */
  int c;      /* character value */
  unsigned n;   /* number of bytes */

Description

  This function, which was introduced with UNIX System V, copies
```

blocks of memory.

Copying stops once either of these conditions is true:

- the specified block size has been copied
- the specified character has been copied.

The memccpy function does not handle overlapping memory blocks. If you specify overlapping blocks to this function, the results are unpredictable.

This function neither recognizes nor produces the NULL terminator byte usually found at the end of strings.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Returns

The memccpy function returns a pointer to the character after the argument c in the from block, or a NULL pointer if the c argument was not found in the first n characters.

See Also

memcpy , strcpy

1.197 memchr()

memchr-Find a character in a memory block

Synopsis

```
#include <string.h>

s = memchr(a,c,n);

void *s;   /* pointer to character in block */
const void *a;   /* block pointers  */
int c;     /* character value */
size_t n;  /* number of bytes */
```

Description

This function finds the first occurence of a character in a block of memory.

This function does not terminate at a NULL byte. It always searches n bytes.

Portability

ANSI

Returns

The memchr function returns a pointer to the first occurrence of the specified character in the block, or a NULL pointer if the character is not found.

1.198 _memcleanup()

_MemCleanup-Deallocate all allocated memory

Synopsis

```
#include <stdlib.h>

void __stdargs _MemCleanup(void);
```

Description

The _MemCleanup function traverses the linked list of allocated memory to release any memory allocated using SAS/C library functions and not yet returned to the system. No cleanup is performed for memory allocated with Amiga operating system calls.

This function is normally called from the SAS/C startup code as a program is terminating. You can replace the standard _MemCleanup function with one of your own.

Portability

AmigaDOS

1.199 memcmp()

memcmp-Compare two memory blocks

Synopsis

```
#include <string.h>

x = memcmp(a,b,n);

int x;       /* return value */
const void *a,*b;  /* block pointers */
size_t n;    /* number of bytes */
```

Description

This function compares two blocks of memory, character by character.

The memcmp function has a built-in version that is equivalent to the standard library versions. A built-in version generates inline 68000 instructions without needing to make calls to the library. The statement #include <string.h> provides a default setting by which any built-in functions are accessed. If you do not want the built-in function, you can use an #undef memcmp statement after including the string.h file.

This function does not terminate at a NULL byte. It always searches n bytes.

Portability

ANSI

Returns

The memcmp function returns an integral value as follows:

```
Return     Meaning
------     -------
negative   first block sorts below the second
zero       first block equals the second
positive   first block sorts above the second
```

1.200 memcpy()

memcpy-Copy a memory block

Synopsis

```
#include <string.h>

s = memcpy(to,from,n);

void *s;      /* return pointer */
void *to;     /* destination pointer */
const void *from;  /* source pointer */
size_t n;     /* number of bytes */
```

Description

This function, which was introduced with UNIX System V, copies blocks of memory.

The memcpy and movmem functions are similar, except the former was introduced with UNIX V, while the latter is a traditional SAS/C function. The memcpy function does not handle overlapping memory blocks. If you specify overlapping blocks to this function, the results are unpredictable. You may want to use the ANSI function memmove instead, since it does handle overlapping blocks.

The memcpy function has a built-in version that is equivalent to the standard library versions. A built-in version generates

```
  inline 68000 instructions without needing to make calls to the
  library. The statement #include <string.h> provides a default
  setting by which any built-in functions are accessed.  If you do
  not want the built-in function, you can use an #undef memcpy
  statement after including the string.h file.

  The memcpy function does not place a NULL byte at the end of the
  block, but it always copies n bytes.
```

Portability

   ANSI

Returns

  The memcpy function returns a pointer to the start of the
  destination block.

See Also

   memccpy , memmove , movmem , strcpy

1.201 memmove()

memmove-Copy bytes in memory

Synopsis

```
  #include <string.h>

  p = memmove(dest, from, nbytes);

  void *p;    /* same as dest           */
  void *dest;   /* destination for moved bytes       */
  const void *from; /* source of bytes for move      */
  size_t nbytes;    /* number of bytes to be transferred */
```

Description

  This function copies the specified number of bytes from one memory
  location to another. It checks the relative addresses supplied to
  determine the direction of transfer that will avoid overlap.

Portability

   ANSI

Returns

  The memmove function returns a pointer to the destination block.

Example

```
  /*
   * Make room to insert a word in a character string.
   *
   * This program produces the following output:
   *    This is a test
   *    This is not a test
   */
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      char string[100];

      strcpy(string,"This is a test");
      printf("%s\n",string);

      /* Shift the words "a test" to make room */
      /* WARNING: Make sure you have plenty of space in */
      /* the area you are working with.  memmove() and  */
      /* others do NOT stop at the terminating NULL of  */
      /* a string, so will blithely write over any      */
      /* memory you tell them to.  This can lead to     */
      /* different types of problems, from simple       */
      /* "strange occurrences" to spectacular crashes!  */

      memmove(string+11,string+7,strlen(string+7)+1);
      memcpy(string+7," not ",5);
      printf("%s\n",string);
  }
```

See Also

   memcpy , movmem , strcpy

1.202 memset()

memset-Set a memory block to a value

Synopsis

```
  #include <string.h>

  s = memset(to,c,n);

  void *s;     /* return pointer */
  void *to;    /* destination pointer */
  int c;       /* character value */
  size_t n;    /* number of bytes */
```

Description

  This function which is compatible with UNIX, sets a block of
  memory to a value.

  The memset function has a built-in version that is equivalent to
  the standard library versions. A built-in version generates

```
  inline 68000 instructions without needing to make calls to the
  library. The statement #include <string.h> provides a default
  setting by which any built-in functions are accessed.  If you do
  not want the built-in function, you can use an

      #undef memset

  statement after including the string.h file.

  This function neither recognizes nor produces the NULL terminator
  byte usually found at the end of strings.

Portability

   ANSI

Returns

  The memset function returns a pointer to the destination block.

See Also

   setmem
```

1.203 min()

```
min-Compute the minimum of two values

Synopsis

  #include <math.h>

  v = min(a,b);

Description

  This macro computes the minimum of two arithmetic values.  It is
  defined as follows:

     #define min(a,b) ((a)<=(b)?(a):(b))

  This macro works with any arithmetic type or combination of types.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   UNIX

Returns

  This macro returns the smallest of the two parameters.

See Also
```

max

1.204 mkdir()

mkdir-Make a new directory

Synopsis

```
  #include <dos.h>

  error = mkdir(path);

  int error;     /* 0 if successful */
  const char *path;  /* points to new directory path string */
```

Description

This function makes a new directory in the specified path. For example, if the path is sys:/abc/def/ghi, then the new directory is named ghi and is in the path /abc/def on the volume labeled sys:. For AmigaDOS, the path may begin with a drive or volume name and a colon.

Portability

UNIX

Returns

If the operation is successful, the function returns 0. Otherwise, it returns -1 and places error information in the external integers errno and _OSERR .

See Also

errno , _OSERR , rmdir

1.205 mkstemp()

mkstemp-Make a unique filename and open the file

Synopsis

```
  #include <unistd.h>

  fh = mkstemp(char *template_arg);

  int fh;    /* file handle */
```

Description

```
  The mkstemp function replaces the contents of the string pointed
  to by template_arg with a unique filename, opens that file for
  reading and writing, and returns a file handle for the file.
  mkstemp prevents any conflict between testing whether the file
  exists and opening the file for use (race conditions).

  The string in template_arg is a filename with embedded X letters.
  mkstemp replaces the Xs with a letter or a digit from the current
  process address, beginning with the low-order digits.  If the
  template does not contain enough Xs to accomodate all of the
  digits in the address, the high-order digits are dropped first.
  The letter is dropped last.

  You can enter as many Xs in the template as you want.


  CAUTION:  Do not use this function with constant strings.  This
  function modifies the content of the buffer sent to it, so any
  constants are changed.  If you compile with the STRMERGE option,
  you could modify your code section.

Portability

   UNIX

Returns

  If successful, this function returns a file handle, which is an
  integer equal to or greater than 0.  If the file could not be
  created, mkstemp returns -1.

Example

  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #include <unistd.h>

  int main(void)
  {
      char buffer[32];
      int fd;
      int rc;

      strcpy(buffer, "TemplateXXXXXXXX");
      fd = mkstemp(buffer);
      if (fd == -1)
      {
    printf("mkstemp() failed!\n");
    rc = EXIT_FAILURE;
      }
      else
      {
    printf("File \"%s\" created.  You should delete it!\n",
           buffer);
    close(fd);
    rc = EXIT_SUCCESS;
      }
      return rc;
  }
```

See Also

```
   mktemp ,  open ,  tmpfile ,  tmpnam
```

1.206 mktemp()

mktemp-Make a unique filename

Synopsis

```
  #include <unistd.h>

  ptr = mktemp(char *template_arg);

  char *ptr;     /* pointer to the template */
```

Description

```
  The mktemp function replaces the contents of the string pointed to
  by template_arg with a unique filename and returns the address of
  template_arg.

  The string in template_arg is a filename with embedded X letters.
  mktemp replaces the Xs with a letter or a digit from the current
  process address, beginning with the low-order digits. If the
  template does not contain enough Xs to accomodate all of the
  digits in the address, the high-order digits are dropped first.
  The letter is dropped last.

  You can enter as many Xs in the template as you want.

  CAUTION:  Do not use this function with constant strings. This
  function modifies the content of the buffer sent to it, so any
  constants are changed.  If you compile with the STRMERGE option,
  you could modify your code section.
```

Portability

```
   UNIX
```

Returns

```
  This function returns a pointer to the template.  If the first
  character in the template is NULL, then mktemp failed to create a
  unique filename.  Otherwise, the name in the template is a unique
  filename.
```

Example

```
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #include <unistd.h>

  int main(void)
  {
      char buffer[32];
      int rc;

      strcpy(buffer, "TemplateXXXXXXXX");
      mktemp(buffer);
      if (buffer[0] == 0)
      {
    printf("mktemp() failed!\n");
    rc = EXIT_FAILURE;
      }
      else
      {
    printf("mktemp() created \"%s\"\n", buffer);
    rc = EXIT_SUCCESS;
      }
      return rc;
  }
```

See Also

mkstemp , tmpfile , tmpnam

1.207 mktime()

mktime-Convert the broken-down time to a time_t value

Synopsis

```
  #include <time.h>

  t = mktime(ts);

  time_t t;   /* number of seconds since 1/1/70 */
  struct tm *ts;    /* broken down time structure */
```

Description

This function converts the broken-down time, expressed as local time, to a time_t value identical to what the time function would return for the specified date and time.

The functions localtime , gmtime , ctime , and mktime share a static data area. A call to any one of these destroys the results of the previous call.

Portability

ANSI

Returns

```
  The mktime function returns the number of seconds since January 1,
  1970.
```

Example

```
  /*
   *
   * Get a time value for a very important event
   *  Sept 8, 1988 20:16:02
   *
   */
  #include <stdio.h>
  #include <time.h>

  void main(void)
  {
      struct tm inputtm;
      time_t    event;

      inputtm.tm_sec  = 02;     /* seconds after the minute   */
      inputtm.tm_min  = 16;     /* minutes after the hour     */
      inputtm.tm_hour = 20;     /* hours since midnight     */
      inputtm.tm_mday = 8;      /* day of the month     */
      inputtm.tm_mon  = 9;      /* months since January     */
      inputtm.tm_year = 88;     /* years since 1900     */
      inputtm.tm_isdst = 1;     /* Daylight Savings Time flag */
      event = mktime(&inputtm);

      printf("%d seconds passed between 1/1/70, 00:00:00"
       " and 9/8/88, 20:16:02\n", event);
  }
```

See Also

```
   time
```

1.208 modf()

modf-Split a floating-point value

Synopsis

```
  #include <math.h>

  x = modf(y,p);

  double x;  /* fractional part of y */
  double y;  /* number to be broken up */
  double *p; /* integral part of y */
```

Description

```
  The modf function separates the integral and fractional parts of
  the argument y and returns them as two double-precision
  floating-point numbers. Both parts have the same sign as the y
  argument. The fractional part is the number that would be obtained
  by calling the  fmod  functions as follows:

      x = fmod(y,1.0);

  Make sure that the second argument of the modf function is a
  pointer to a double-precision floating-point number. Do not use a
  pointer to an integer.
```

Portability

   ANSI

Returns

```
  The function return value is the fractional part of the argument
  y, and the integral part is placed in the double-precision
  floating-point number pointed to by the argument p.
```

Example

```
  #include <stdio.h>
  #include <math.h>

  void modit(double fi)
  {
      double ff;

      ff = modf(1.2,&fi);        /* ff contains 0.2 */
      printf("modf(1.2,%lf) = %lf\n", fi, ff);
  }

  void main(void)
  {
      modit(1.0);
  }
```

See Also

   fmod

1.209 movmem()

movmem-Move a memory block

Synopsis

```
  #include <string.h>

  void movmem(from,to,n);

  const void *from;  /* source pointer */
  void *to;    /* destination pointer */
  unsigned n;    /* number of bytes */
```

Description

```
  This function moves blocks of memory. The movmem function handles
  overlapping memory blocks correctly.

  This function does not terminate on a NULL byte.  It always moves
  exactly n bytes.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

```
   UNIX
```

See Also

```
   memmove
```

1.210 mrand48()

mrand48-Generate a random long integer (internal seed)

Synopsis

```
  #include <math.h>

  z = mrand48(void);

  long z;        /* random long */
```

Description

```
  This function generates random numbers using the linear
  congruential algorithm and 48-bit arithmetic.  The mrand48
  function uses an internal 48-bit storage area for the seed value.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

```
   UNIX
```

Returns

```
  The mrand48 function returns signed long integers uniformly
  distributed over the interval from -2**31 to 2**31-1.
```

See Also

```
   jrand48 ,  lcong48 ,  nrand48 ,  rand ,  srand
```

1.211 nrand48()

nrand48-Generate a random positive long integer (external seed)

Synopsis

```
  #include <math.h>

  y = nrand48(seed);

  long y;     /* random positive long */
  unsigned short seed[3]; /* seed value (high bits in seed[0]) */
```

Description

This function generates random numbers using the linear congruential algorithm and 48-bit arithmetic. The nrand48 function is provided for cases where several seeds are in use at the same time, so you can specify the seed on each function call.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

UNIX

Returns

The nrand48 function returns nonnegative long integers uniformly distributed over the interval from 0 to 2**31-1.

See Also

jrand48 , lcong48 , lrand48 , mrand48 , rand , srand

1.212 offsetof()

offsetof-Get the byte offset of a structure member

Synopsis

```
  #include <stddef.h>

  size_t offsetof(type, element);
```

Description

The offsetof macro returns a size_t constant specifying the decimal byte offset of a component within a structure. This constant is generated at compile time. Padding for alignment, if any, is included. The operands of the offsetof function are a structure type (type) and structure member (element). The element does not include the structure type or the selection operators '.'

```
  or '->'.

Portability

   ANSI

Returns

  The offsetof function returns the byte offset of element, within
  the structure type.

Example

  This section contains several examples of the use of the offsetof
  macro. In these examples, the member specification is written as
  it would be written to access the value of a structure member,
  except that there is no leading '.' or '->' selection operator.

  #include <stddef.h>

  struct AAA {  /* Define structure AAA */
      double ddd;
      char ccc;
      int bbb;
  };

  size_t x;

  /* x is the byte offset of component bbb in */
  /* struct AAA */
  x = offsetof(struct AAA, bbb);


  The following example shows a structure, data, with an inner
  structure base.

  #include <stddef.h>

  struct data { /* Define struct data */
      int id;
      int *elem;
      char *name;
      struct {  /* Define struct type base */
    double proj;
      } base;
  };

  long ofs;

  /* ofs is the byte offset of base.proj */
  ofs = offsetof(struct data, base.proj);


  In the following example, complex is defined with a typedef
  statement to be a structure type.  The component specification
  inner.d[5] specifies an array element within an inner structure.
  The variable y is set to the offset of the sixth array element in
  the inner structure (decimal 56).

  #include <stddef.h>

  typedef struct {  /* Define struct type complex */
      struct XXX *xptr, *xptr2;
      struct {    /* Define struct type inner */
    int count, count2;
    double d[10];
      } inner;
      struct XXX *xptr3;
  } complex;

  /* y is the byte offset of inner.d[5] */
  long y;
  y = offsetof(complex, inner.d[5]);
```

1.213 onbreak()

onbreak-Plant a break trap

Synopsis

```
  #include <dos.h>

  error = onbreak(func);

  int error;     /* 0 if successful */
  int (*func)(void); /* pointer to function to be called */
```

Description

This function plants a break trap, which is simply a function that gets called whenever the user enters Control-C or Control-D. The standard break keys Control-E and Control-F are ignored by the onbreak function.

The onbreak function can perform any AmigaDOS operations. If it returns a value of 0, then execution resumes at the interrupted point. Otherwise, the program is aborted immediately.

If the func argument is NULL, then the current break trap, if any, is removed and the default interrupt handler is restored. With the default handler, Control-C and Control-D cause a requester to appear on the screen with choices to continue or abort.

Detection of Control-C and Control-D is performed during level 1 I/O. Explicit checks for these events can be forced by calls to the function chkabort .

Portability

AmigaDOS

Returns

```
  The onbreak function returns 0 if it was successful.  The break
  trap function should return 0 to continue execution and a nonzero
  value to abort.
```

Example

```
  This program tests the onbreak function.  After the initial
  message is printed, you should get the Break received message when
  you press Control-C or Control-D.  The second time causes the
  program to terminate.

  #include <stdio.h>
  #include <dos.h>

  int i = 0;

  int brk(void)            /* This is the break function */
  {
      printf("Break received...\n");
      return(i++);
  }

  void main(void)          /* This is the main program  */
  {
      printf("Setting break trap...\n");
      if (onbreak(&brk))
      {
    printf("Can't set break trap\n");
      }
      while(1)
      {
    chkabort();    /* check for CTRL-C          */
      }
  }
```

See Also

chkabort , signal

1.214 onexit()

onexit-Set an exit trap

Synopsis

```
  #include <stdlib.h>

  success = onexit(func);

  int success;          /* non-zero for success */
  void (*func)(void);    /* trap function pointer */
```

Description

This function establishes an exit trap (function) that is called when the program terminates. The trap function is called just before the program returns to the operating system. All buffers are flushed and files are closed before the trap is called. User-allocated memory is not yet freed.

This implementation of the onexit function allows only one trap. Each call to the onexit function overrides the previous trap. If you call the onexit function with a NULL pointer, the current trap is removed. (You can use the atexit function instead. atexit allows more than one trap and is an ANSI standard function.)

CAUTION: The exit trap is called after all files have been closed. Do not call the printf or vprintf function from within the exit trap.

Portability

OLD

Returns

If the function is successful, a nonzero value is returned.

Example

```
/* This program tests the onexit function. */

#include <stdlib.h>
#include <stdio.h>

int ex(int i)   /* This is the exit trap function */
{
    Write(Output(),"Exit trap hit\n",14);
    return(0);
}

void main(void)        /* This tests the exit trap */
{
    printf("Setting exit trap...\n");
    if (!onexit(ex))
  printf("Can’t set trap...\n");
    printf("Exiting with code 2\n");
    exit(2);
}
```

See Also

atexit , exit

1.215 open()

open-Open a level 1 file

Synopsis

```
  #include <fcntl.h>

  fh = open(name,mode,prot);

  int fh;        /* file handle */
  const char *name;    /* file name */
  int mode;      /* access mode */
  int prot;      /* protection mode (O_CREAT only) */
```

Description

This function opens a file so that it can be accessed with the level 1 I/O functions. The filename can be any character string that is a valid filename, and it may include a device code and a directory path. The access mode is formed by combining the appropriate symbols using the logical OR operator (|) from the following list of conventional UNIX symbols:

O_RDONLY
specifies read-only access. No writes are allowed.

O_WRONLY
specifies write-only access. No reads are allowed.

O_RDWR
specifies read-write access. Both reads and writes are allowed.

O_NDELAY
is defined for UNIX compatibility and has no effect under AmigaDOS.

O_APPEND
is normally used in conjunction with the O_WRONLY or O_RDWR symbols. It causes the I/O system to seek to the end of the file before the first write operation.

O_TRUNC
specifies that if the file exists, it is truncated to a length of 0. This flag is normally used with the O_CREAT, O_WRONLY or O_RDWR symbol.

O_CREAT
specifies that if the file does not already exist, it is created. The protection mode argument is provided for compatibility with existing software, but is ignored under AmigaDOS. To set the protection use the chmod function to change the protection bits after the file has been closed.

O_EXCL
is used only with the O_CREAT symbol. If the O_EXCL and O_CREAT symbols are both present and the file already exists, the open function will fail.

The prot parameter is only required if the mode is equal to

```
  O_CREAT.  The protection bits are defined in the file dos/dos.h.
```

Portability

```
   UNIX
```

Returns

```
  If the operation is successful, the function returns a file
  handle, which is an integer equal to or greater than 0.
  Otherwise, it returns -1 and places error information in the
  external integers  errno  and  _OSERR .
```

See Also

```
   chmod ,  close ,  creat ,  errno ,  _OSERR
```

1.216 opendir()

opendir-Initiate a directory operation

Synopsis

```
  #include <sys/dir.h>

  dfd = opendir(dirname);

  DIR  *dfd;    /* return directory file descriptor */
  const char *dirname;  /* directory name */
```

Description

```
  Given a directory name, this routine opens the directory for read
  access.
```

Portability

```
   UNIX
```

Returns

```
  If successful, this function returns a pointer to a handle that
  contains the following information:

      typedef struct _dirdesc {
    long  dd_fd;  /* system directory lock   */
    long  dd_loc; /* current directory posn  */
    long  dd_size;  /* size of dd_buf in bytes */
    char *dd_buf; /* system structure info   */
      } DIR;

  If it is not successful, this function returns a NULL pointer.
```

Example

```
/* An example of opening and searching the contents */
/* of a directory for a particular entry.     */
#include <sys/dir.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int searchdir(char *dname, char *file)
{
    int rc;
    DIR *dfd;     /* directory descriptor */
    struct dirent *dptr;  /* dir entry    */

    rc = 0;
    dfd = opendir(dname);
    while ((dptr = readdir(dfd)) != NULL)
    {
  if (!stricmp(file, dptr->d_name))
  {
      rc = 1; /* Found a match */
      break;
  }
    }
    closedir(dfd);
    return(rc);
}

void main(int argc, char *argv[])
{
    if (argc < 3)
    {
  printf("Usage: OpenDir <dirname> <filename>\n");
  exit(EXIT_FAILURE);
    }
    if (searchdir(argv[1], argv[2]) == 0)
    {
  printf("File \"%s\" not found in " "directory \"%s\"!\n",
         argv[2], argv[1]);
  exit(EXIT_FAILURE);
    }
    printf("Found it!\n");
}
```

See Also

closedir , readdir , seekdir , telldir

1.217 ovlymgr()

ovlyMgr-Overlay manager call point

Synopsis

```
  MOVEQ #ovent,D0
  JMP   ovlyMgr
```

Description

This function is the main entry point to the overlay manager that is called whenever control is to be transferred to an alternate overlay node. The input parameter is an index into the overlay ordinate table that is constructed by the linker. These offsets are automatically assigned by the linker, and all calls to routines across overlay node boundaries are automatically routed through the overlay manager entry point.

Source code for the overlay manager is found in the ovs.a file in the source directory.

Calls through the overlay manager destroy registers D0, D1, A0, and A1. Therefore, registerized parameters do not work, and all calls through the overlay manager should use standard stack-based parameter passing (__stdargs).

Portability

AmigaDOS

1.218 perror()

perror-Print error message

Synopsis

```
  #include <stdio.h>

  void perror(s);

  const char *s;   /* message prefix */
```

Description

This function checks the external integer errno and, if it is nonzero, sends an error message to stderr. The message consists of the specified prefix, a colon, a space, and the message text from the external array named __sys_errlist . This array contains pointers to the various UNIX error messages. The highest error number is given by the contents of the external integer __sys_nerr . The SAS/C Compiler software contains the source for these two external items in a file named syserr that allows you to modify the messages. The __sys_nerr external integer and the __sys_errlist external array are declared in the header file errno .h.

Portability

ANSI

See Also

   errno , poserr , __sys_errlist , __sys_nerr

1.219 poserr()

poserr-Print AmigaDOS error message

Synopsis

```
  #include <dos.h>

  error = poserr(s);

  int error;  /* contents of  _OSERR  */
  const char *s;  /* message prefix */
```

Description

  This function checks the external integer _OSERR and, if it is nonzero, sends an error message to stderr. The message consists of the specified prefix, a colon and space, and the message text from the external array named __os_errlist . This array of structures contains pointers to the various AmigaDOS error messages. The highest error number is given by the contents of the external integer __os_nerr . The __os_errlist external array and the __os_nerr external integer are declared in the file errno .h. The SAS/C Compiler software contains the source for these two external items in a file named oserr.c, which you can modify to customize or expand the messages.

Portability

   AmigaDOS

Returns

  The function returns the contents of the external integer _OSERR so you can test for an error condition and print a message in one step, as in the example:

```
      if (poserr("foo")) goto abort;
```

See Also

   _OSERR , perror

1.220 printf()

printf-Formatted print to stdout

Synopsis

```
#include <stdio.h>

length = printf(fmt,arg1,arg2,...);

int length;    /* number of characters generated */
const char *fmt;   /* format string */
.... argx;      /* arguments */
```

Description

This function produces an output stream of ASCII characters and sends the output to stdout. stdout is usually the user’s screen (console).

The printf function has a built-in version that is equivalent to the standard library version. The effect of a call to the printf function is that the most efficient internal version of the printf function is used.

This function works like the fprintf function. See the description of the fprintf function earlier in this chapter for a complete description.

Portability

ANSI

Returns

This function returns the number of output characters generated.

Example

```
/*
 * This example prints a message indicating whether
 * the function argument is positive or negative.
 * In the second printf, the width and precision
 * are 15 and 8, respectively.
 */
#include <stdio.h>

void pneg(double value)
{
    char *sign;

    if (value < 0)
    {
  sign = "negative";
    }
    else
    {
  sign = "not negative";
    }

    printf("The number %E is %s.\n",value,sign);
    printf("The number %*.*E is %s.\n",15,8,value,sign);
}

void main(void)
{
    pneg(37.8);
    pneg(-18.2);
}
```

See Also

   fprintf

1.221 pow()

pow-Raise a number to a power

Synopsis

```
  #include <math.h>

  r = pow(x,y);

  double r, x, y;
```

Description

  The pow function raises the argument x to the y power. If x is
  negative and y is not an integral value, the __matherr function is
  called with a DOMAIN error.

Portability

   ANSI

See Also

   __matherr , pow2

1.222 pow2()

pow2-Raise 2 to a power

Synopsis

```
  #include <math.h>

  r = pow2(x);

  double r, x;
```

Description

The pow2 function computes 2**x by calling the pow function.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Returns

The return value r is the value 2**x.

See Also

__matherr

1.223 _prolog()

_PROLOG-Profiler _PROLOG hook

Synopsis

```
#include <sprof.h>

void __asm _PROLOG(register __a0 const char *where);
```

Description

If you compile a function with the profile option, _PROLOG and _EPILOG are automatically called when the function is entered or exited, respectively. _PROLOG and _EPILOG were designed for use by the sprof utility, but you can replace them with your own code and use them for any purpose. The SAS/C versions of _PROLOG and _EPILOG note the time the function was entered and exited and pass this information to the sprof utility, which produces a report telling you how much time was spent in each function.

If you declare _PROLOG and _EPILOG in your code, make sure you declare them with the __asm and register __a0 keywords as shown. If you declare either _PROLOG or _EPILOG , you must declare both, even if one of them simply returns immediately.\nl sc:source/profile.c contains the source code for the SAS/C versions of _PROLOG and _EPILOG and the autoinitialization and autotermination functions associated with them.

The parameter where is passed on the stack. It points to a character string of the following form:

```
"\module\function\line"
```

where

module
is the name of the file containing the function

```
      function
    is the name of the function

      line
    is the line number on which the function begins.

  For example, if you have a function foo beginning on line 17 of
  the file foo.c, the where parameter would be

      "\foo.c\foo\17"

  A null where parameter indicates that the PROFILER_ON or
  PROFILER_OFF macro has been called. PROFILE_OFF turns off
  profiling for the code that follows it. PROFILE_ON reinstates
  profiling.

  For more information about profiling, _PROLOG,  _EPILOG ,
  PROFILE_ON, and PROFILE_OFF, refer to the description of the sprof
  utility in SAS/C Development System User's Guide, Volume 2.
```

Portability

```
   SAS/C
```

See Also

```
   _EPILOG
```

1.224 putc()

putc-Put a character to a level 2 file

Synopsis

```
  #include <stdio.h>

  r = putc(c,fp);

  int r;      /* EOF or c */
  int c;      /* Character to be output */
  FILE *fp;   /* Level 2 file pointer */
```

Description

```
  This function puts a single character to the specified level 2
  file.  The putc function is implemented as a macro to maximize
  execution speed.  Therefore, you should not pass expressions that
  may have side effects to the putc function.  For example,
  putc(c++,fp) may increment c more than once.
```

Portability

```
   ANSI
```

```
Returns

  If successful, this function returns the character to be output;
  otherwise, it returns EOF.  For disk files, an EOF return usually
  means that the disk is full.  However, this type of return can
  also occur if the device is write-protected or if a write error
  occurs.  In any case, additional error information can be found in
  the external integers  errno  and  _OSERR .

See Also

   errno ,  fopen ,  _OSERR
```

1.225 putchar()

```
putchar-Put a character to stdout

Synopsis

  #include <stdio.h>

  r = putchar(c);

  int r;      /* EOF or c */
  int c;      /* Character to be output */

Description

  This function puts a single character to the level 2 file stdout.
  The putchar function is implemented as a macro to maximize
  execution speed. Therefore, you should not pass expressions that
  may have side effects to the putchar function.  For example,
  putchar(c++) may increment c more than once.

Portability

   ANSI

Returns

  If successful, this function returns the character that was
  output; otherwise, it returns EOF.  For disk files, an EOF return
  usually means that the disk is full.  However, this type of return
  can also occur if the device is write-protected or if a write
  error occurs.  In either case, additional error information can be
  found in the external integers  errno  and  _OSERR .

See Also

   errno ,  fopen ,  _OSERR
```

1.226 putenv()

```
putenv-Put a string into the environment

Synopsis

  #include <stdlib.h>

  error = putenv(env);

  int error;      /* 0 if successful  */
  const char *env;   /* environment string */

Description

  The putenv function accepts a string and places it into the
  current environment.  This string has the form:

      name=var

  If the environment already contains a string beginning with the
  specific name then that string is replaced; otherwise, the new
  string is added.  The text of var is written into the file

      ENV:name.

  Environment variables on the Amiga are global so that writing an
  environment variable from one process sets the variable for all
  processes.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   UNIX

Returns

  If the putenv function is unable to write to the file ENV:name, it
  returns a nonzero return code.

Example

  #include <stdio.h>
  #include <stdlib.h>

  void main(void)
  {
      char *e;

      if (putenv("HOCUS=pocus"))        /* Add HOCUS */
      {
    printf("Couldn't add HOCUS\n");
    exit(EXIT_FAILURE);
      }
      printf("Environment variable HOCUS added\n");
      e = getenv("HOCUS");              /* Read HOCUS */
      if (e == NULL)
      {
    printf("Couldn't read HOCUS\n");
    exit(EXIT_FAILURE);
      }
      printf("Environment variable HOCUS contains: %s\n", e);

      printf("Removing environment variable HOCUS\n");
      if (putenv("HOCUS="))             /* Remove HOCUS */
      {
    printf("Couldn't remove HOCUS\n");
    exit(EXIT_FAILURE);
      }
      printf("Done.\n");
  }
```

See Also

   getenv

1.227 putreg()

putreg-Set up 680x0-specific registers

Synopsis

```
  #include <dos.h>

  void putreg(reg,x);

  int reg;  /* register number */
  long x;   /* value of reg */
```

Description

The built-in function putreg assigns a value to a register. The valid register values (for the reg argument) are defined in the file dos.h as follows:

```
        Register        Register    Register
      Value   Name       Value  Name   Value  Name
      -----   -------- | -----  -------- | -----  --------
        0     REG_D0   |   8    REG_A0   |  16  REG_FP0
        1     REG_D1   |   9    REG_A1   |  17  REG_FP1
        2     REG_D2   |  10    REG_A2   |  18  REG_FP2
        3     REG_D3   |  11    REG_A3   |  19  REG_FP3
        4     REG_D4   |  12    REG_A4   |  20  REG_FP4
        5     REG_D5   |  13    REG_A5   |  21  REG_FP5
        6     REG_D6   |  14    REG_A6   |  22  REG_FP6
        7     REG_D7   |  15    REG_A7   |  23  REG_FP7
```

The floating-point registers FP0 through FP7 are available only on Amigas with a math coprocessor. Therefore, you will get an error if you attempt to refer to FP0 through FP7 when the math=68881 compiler option is active.

```
  CAUTION:  Incorrect use of this function can cause serious
  problems.
```

Portability

   SAS/C

Example

```
  putreg(REG_A4,x); /* set up the current global static base */
```

See Also

   getreg

1.228 puts()

puts-Put string to stdout

Synopsis

```
  #include <stdio.h>

  error = puts(s);

  int error;   /*non-zero if error*/
  const char *s;   /*string pointer*/
```

Description

  The puts function copies string s to stdout (the standard output
  file). The terminating NULL byte is not copied, but a new line
  character (\n) is sent after the string.

Portability

   ANSI

Returns

  If an error occurs, the return value is -1; otherwise, it is 0.
  Additional error information can be found in the external integers
   errno and _OSERR .

Example

```
  /*
   * This examples writes the following two lines to stdout:
   *
   * This is the first line
   * This is the second line
   */
  #include <stdio.h>
  void main(void)
  {
      puts("This is the first line");
      fputs("This is ",stdout);
      puts("the second line");
  }
```

See Also

```
   errno ,  ferror ,  fopen ,  fputc ,  fputs
```

1.229 qsort()

qsort-Sort a data array

Synopsis

```
  #include <stdlib.h>

  void qsort(a,n,size,cmp);

  void *a;          /* data array pointer */
  size_t n;         /* number of elements in array */
  size_t size;          /* element size in bytes */
     /* pointer to comparison function */
  int (*cmp)(const void *, const void *);
```

Description

```
  The qsort function sorts the specified array using the ACM 271
  algorithm, more popularly known as Quicksort.  During its
  operation, it calls upon the specified comparison routine with
  pointers to the two array elements being compared.  The comparison
  routine should return an integral result as follows:

      Return    Meaning
      ------    -------
      negative  first element is below the second
      zero      elements are equal
      positive  first element is above the second
```

Portability

```
   ANSI
```

1.230 raise()

raise-Generate a signal

Synopsis

```
  #include <signal.h>
  ret = raise(sig);

  int ret;   /* 0 if successful, nonzero if failed */
  int sig;   /* signal to generate           */
```

Description

  This function simulates the generation of a signal and invokes the
  proper signal handler for that signal.

Portability

   ANSI

Returns

  A nonzero return value indicates failure.

Example

```
  /*
   * Cause a floating point overflow
   */
  #include <signal.h>

  void main(void)
  {
      raise(SIGFPE);
  }
```

See Also

   __matherr , onbreak , signal

1.231 rand()

rand-Generate a random number

Synopsis

```
  #include <stdlib.h>

  x = rand(void);

  int x;     /*  random number */
```

Description

  The rand function returns pseudo-random numbers in the range from
  0 to the maximum positive integer value (RAND_MAX).

  See the drand48 function and its related functions for more
  sophisticated random number generation.

Portability

   ANSI

Returns

  This function returns an integer value as noted above.

Example

```
/*
 * This example prints 1000 random numbers
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void main(int argc, char *argv[])
{
    int i, x;

    if (argc > 1)
    {
  stcd_i(argv[1],&x);
  if (x == 0)
  {
      x = 1;
  }
    }
    else
    {
  x=time(NULL);
    }
    srand(x);
    printf("Seed value is %d\n",x);
    printf("Here are 1000 random numbers...\n");
    for(i = 0; i < 200; i++)
    {
  printf("%5d %5d %5d %5d %5d\n",
         rand(),rand(),rand(),rand(),rand());
    }
    printf("\n\n");
}
```

See Also

   drand48 , srand

1.232 rawcon()

rawcon-Set or unset raw console input

Synopsis

```
#include <stdio.h>
```

```
  error = rawcon(flag);

  int error;    /* 0 on success */
  int flag;   /* non-zero for raw, 0 for non-raw */
```

Description

  This routine turns on and off the capability of stdin that allows
  you to get single character input without waiting for a new line
  character.

  This routine works with the getch and getchar functions.
  Normally, the Amiga console device waits until you press the Return
  key before it passes the keystrokes you enter to your program.
  Therefore, the getchar function, for example, will not be able to
  read a single character at a time. Calling rawcon(1) forces the
  console device to pass each character separately. Calling the
  rawcon(0) function restores the console to normal operations.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   AmigaDOS

Returns

  A nonzero return value indicates failure.

Example

```
  /*
   *
   * Wait for user to press any key (if possible)
   *
   */
  #include <stdio.h>
  #include <stdlib.h>

  void main(void)
  {
      if (!rawcon(1))
      {
    printf("Press any key to continue\n");

    /* make sure output from printf is seen */
    fflush(stdout);
    getchar();
    rawcon(0);
      }
      else
      {
    /* unable to switch the console, wait some other way */
    printf("Sorry, rawcon() didn’t work!\n");
    exit(EXIT_FAILURE);
```

```
      }
  }
```

See Also

   getch , getchar

1.233 rbrk()

rbrk-Release memory

Synopsis

```
  #include <stdlib.h>

  error = rbrk(void);

  int error;    /* 0 if successful */
```

Description

  The rbrk function returns all memory in the memory pool, including
  level 2 I/O buffers, to the operating system.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   OLD

Returns

  This function returns 0 if successful.

See Also

   getmem , lsbrk , malloc , sbrk

1.234 read()

read-Read from a level 1 file

Synopsis

```
  #include <fcntl.h>

  count = read(fh,buffer,length);

  int count;        /* actual bytes read or written */
  int fh;         /* file handle */
  void *buffer;       /* data buffer */
  unsigned int length;  /* number of bytes to read or write */
```

Description

This function reads a level 1 file whose handle was returned by the creat or open function. Under normal circumstances, the value returned should match the buffer length. If this value is -1 or greater than the requested length, then some type of errorete()}, ←
scr_cinsert , scr_clear ,
scr_cr , scr_curs , scr_cursrt , scr_cursup , scr_eol ,
scr_home , scr_ldelete , scr_lf , scr_linsert , Console control sequences in Amiga ROM Kernel Reference Manual: Devices

1.235 seed48()

seed48-Set all 48 bits of an internal seed

Synopsis

```
  #include <math.h>

  pseed = seed48(seed);

  unsigned short seed[3]; /* seed value (high bits in seed[0]) */
  unsigned short *pseed;  /* pointer to internal seed */
```

Description

This function generates random numbers using the linear congruential algorithm and 48-bit arithmetic.

The seed48 function allows you to initialize the internal 48-bit seed to something other than the default. The entire 48 bits are loaded from the specified array.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

UNIX

Returns

This function returns a pointer to the internal seed array.

See Also

lcong48 , rand , srand , srand48

1.236 seekdir()

```
seekdir-Reposition a directory operation

Synopsis


  #include <sys/dir.h>

  void seekdir(dfd, loc);
  DIR *dfd;   /* dir file descriptor */
  long loc;   /* entry location */

Description

  This routine changes the position from which the  readdir  function
  will read the next directory entry.  The dfd argument is a
  directory file descriptor returned from a successful call to the
   opendir  function.  The loc argument is a directory entry location
  as returned from a call to the  telldir  function.

  Seek locations are guaranteed only for the life of the directory
  file descriptor.  If a directory is closed and reopened, the
  directory entry locations may be invalid.

Portability

   UNIX

See Also

   closedir ,  opendir ,  readdir ,  rewinddir
```

1.237 setbuf()

```
setbuf-Set the buffer mode for a level 2 file

Synopsis

  #include <stdio.h>

  void setbuf(fp,buff);

  FILE *fp;     /*  file pointer */
  const char *buff;   /*  buffer pointer */

Description

  This function sets the buffering mode for a level 2 file.  You
  should call this function after calling the  fopen  function and
  before calling any other level 2 I/O functions.  If you do not
  call this function in the correct sequence, the file may become
  corrupted.  Do not allocate a buffer on the stack within a
  function, attach it to a file, and then return from the function.
  This will corrupt the stack and cause a system failure.
```

The level 2 I/O system automatically allocates a buffer using the getmem function when you perform the first read or write operation. Then, the data being read or written are staged through this buffer to improve I/O efficiency. If you would rather use your own buffer instead of having one allocated for you, call the setbuf function with a non-NULL buffer pointer. The buffer size must be at least as large as the value given in the external integer __buffsize , which defaults to the value of the symbol BUFSIZ, defined in the file stdio.h.

You can eliminate buffered I/O by calling the setbuf function with a NULL buffer pointer. When this is done, physical I/O occurs whenever your program performs a level 2 read or write operation, even if only one byte is being transferred. This is very inefficient for disk files but often desirable for terminal or communication ports.

Portability

ANSI

See Also

fopen , setnbf , setvbuf

1.238 setjmp()

setjmp-Set long jump parameters

Synopsis

```
#include <setjmp.h>

ret = setjmp(save);

int ret;    /* return code */
jmp_buf save;   /* address of save area */
```

Description

The setjmp function saves the current stack mark in a specified save area and returns a code of 0. A subsequent call to the longjmp function with the same save area causes control to return to the next statement after the original setjmp call. In other words, the statement immediately after the setjmp call will be executed twice, once after you call the setjmp function and once after you call the longjmp function.

Do not use the longjmp function after the function calling setjmp has returned to its caller. This cannot possibly succeed, because the stack frame for that function no longer exists.

This mechanism is useful for quickly popping up through multiple layers of function calls under exceptional circumstances.

```
  From within a shared library, you must not call any library
  functions that terminate your program.  For example, you cannot
  call  exit ,  __exit , or  abort  from a shared library.  You also
  cannot use setjmp and  longjmp  to jump across a call from the
  program into the library.
```

```
Portability
```

```
   ANSI
```

```
Returns
```

```
  A return code of 0 from the setjmp function indicates that this is
  the initial call to save the stack.  A nonzero return code
  indicates that the  longjmp  function has been executed.
```

```
Example
```

```
  #include <stdio.h>
  #include <setjmp.h>

  jmp_buf save;

  void foo(void)
  {
      longjmp(save, 1);
  }

  void main(void)
  {
      int ret;

      ret=setjmp(save);

      if (ret==0)
      {
    /* setjmp has been called, but not longjmp */
    foo();
      }
      else
      {
    /* longjmp has been called */
    printf("all done\n");
      }
  }
```

```
See Also
```

```
   longjmp
```

1.239 setlocale()

```
setlocale-Set locale information for a program
```

Synopsis

```
  #include <locale.h>

  ret = setlocale(category, locale);

  char *ret;       /* Pointer to the selected locale portion */
  int category;      /* Names the portion of the locale to be  */
          /*   selected             */
  const char *locale;  /* Identifies the type of environment     */
```

Description

```
  This function selects the appropriate portion of the program's
  locale as specified by the category and locale arguments. The
  category argument indicates which portion of a program's locale
  will be affected.  You must specify one of the following values:

      Value Portion Affected
      ----- ----------------
      LC_COLLATE  the behavior of the  strcoll  and  strxfrm  functions
      LC_CTYPE  the character-handling and multibyte functions
      LC_NUMERIC  the decimal point character for the formatted I/O
      and string conversion functions, and the
      nonmonetary formatting information returned by the
       localeconv  function
      LC_TIME the behavior of the  strftime  function
      LC_MONETARY the monetary formatting information returned by
      the  localeconv  function

      LC_ALL  the entire program's locale

  The locale string, which identifies the type of environment to
  use, may contain one of three special values:

      Value   Meaning
      -----   -------
       C      Use the minimal environment for C translation.
       ""     Use the Amiga native environment.
       NULL   Use the current default locale without changing it.

  If the locale argument is not one of these strings, the setlocale
  function searches its internal list of locale environments for a
  matching one.  If it finds one, it uses it. Otherwise, it attempts
  to open a disk-based locale specification by using the  readlocale
  function.
```

Portability

```
   ANSI
```

Returns

```
  If it finds the selected environment, the setlocale function
  returns a pointer to a string associated with the requested
  category.  If it cannot find the environment, it returns a NULL
  pointer, and the program's locale is not changed.  This string is
  considered read-only and is valid until the next call to the
  setlocale function.
```

See Also

```
   localeconv ,  readlocale
```

1.240 setmem()

setmem-Set a memory block to a value

Synopsis

```
  #include <string.h>

  void setmem(to,n,c);

  void *to;      /* destination pointer */
  unsigned n;    /* number of bytes */
  int c;         /* character value */
```

Description

  This function sets blocks of memory to a value. It neither
  recognizes nor produces the NULL terminator byte usually found at
  the end of strings.

  This function is similar to the built-in function memset , which is
  an ANSI function.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   UNIX

See Also

```
   memset ,  repmem
```

1.241 setnbf()

setnbf-Turn off I/O buffering for level 2 file

Synopsis

```
  #include <stdio.h>

  void setnbf(fp);

  FILE *fp;   /* file pointer */
```

Description

```
  This function eliminates buffered I/O for a level 2 file.  You
  should call this function after calling the  fopen  function and
  before calling any other level 2 I/O functions.  If you do not
  call this function in the proper sequence, the file may become
  corrupted.

  After you call the setnbf function, physical I/O occurs whenever
  your program performs a level 2 read or write operation, even if
  only one byte is being transferred. This is very inefficient for
  disk files but often desirable for terminal or communication
  ports.

  This function is equivalent to:

      setbuf(fp,NULL);

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

   UNIX

See Also

   fopen , setbuf , setvbuf

1.242 setvbuf()

setvbuf-Set the buffer mode for a level 2 file

Synopsis

```
  #include <stdio.h>

  error = setvbuf(fp,buff,type,size);

  int error;        /*  0 if successful */
  FILE *fp;       /*  file pointer */
  const char *buff;     /*  buffer pointer */
  int type;       /*  type of buffering */
  size_t size;        /*  buffer size in bytes */
```

Description

```
  This function sets the buffering mode for a level 2 file.  You
  should call this function after calling the  fopen  function and
  before calling any other level 2 I/O functions. If you do not call
  this function in the correct sequence, the file may become
  corrupted.  Do not allocate a buffer on the stack within a
  function, attach it to a file, and then return from this function.
  This sequence will corrupt the stack and cause a system failure.
```

The level 2 I/O system automatically allocates a buffer using the getmem function when you perform the first read or write operation. Then the data being read or written are staged through this buffer to improve I/O efficiency. If you would rather use your own buffer instead of having one allocated for you, call the setvbuf function with a non-NULL buffer pointer. The buffer size must be at least as large as the value given in the external integer __buffsize , which defaults to the value of the symbol BUFSIZ, defined in the file stdio.h.

The setvbuf function can set line-buffered mode, attach a buffer of nonstandard size, or turn off buffering. The type argument must be one of the following symbols defined in the file stdio.h:

```
Type    Meaning
----    -------
_IOFBF  fully buffered
_IOLBF  line buffered
_IONBF  nonbuffered
```

For the _IOFBF and _IOLBF symbols, the specified buffer is attached to the file unless the buff argument is NULL, in which case a buffer is automatically allocated on the first read or write. For the _IONBF symbol, the buff and size arguments are ignored.

The line-buffered mode is useful for interactive applications. When in this mode, the buffer is flushed whenever a new line is sent, the buffer is full, or input is requested. However, you must use the fputc and fputchar functions instead of the putc and putchar macros for line buffering to work correctly. The macros do not check if line-buffered mode is active, so they behave as if the file were fully buffered.

Portability

ANSI

Returns

The setvbuf function returns a nonzero error code if type or size is invalid.

See Also

fopen , setbuf , setnbf

1.243 signal()

signal-Establish event traps

Synopsis

```
#include <signal.h>
```

```
  oldfun = signal(sig,newfun);

  void (*oldfun)(int);  /* old trap function */
  int sig;        /* signal number */
  void (*newfun)(int);  /* new trap function */
```

Description

```
  This function establishes traps for various events that can occur
  outside of your program.

  The newfun argument specifies the action to be taken when the
  signal occurs, as follows:

      SIG_IGN
    ignore the signal.

      SIG_DFL
    take the system default action, as indicated above for
    each signal.

  If the newfun argument is not either of the above, then it must be
  a valid function pointer. When the signal is detected, the action
  is reset to either SIG_DFL or SIG_IGN, depending on the particular
  signal.  Then, the trap function is called with an integer
  argument specifying which signal was detected (for example,
  SIGINT).  The trap function can take whatever action is necessary,
  including calling the signal function again to re-establish itself
  as the trap function.  If the function returns, execution
  continues at the point in your program where the signal was
  detected.

  The sig argument specifies which signal is being trapped, using
  the following symbols defined in the file signal.h:

      SIGFPE
    occurs whenever a floating-point error is detected and the
    standard version of the CXFERR function is installed.  If
    you install your own version, you must duplicate our code
    (supplied as a file named CXFERR.C) to provide this
    signal.

      SIGINT
    occurs whenever the user operates the Control-C or
    Control-D key combination.  The default action for
    AmigaDOS is to abort your program.  If you specify a
    function to be called, the signal is reset to SIG_IGN when
    the interrupt occurs.  Your function should call the
    signal function again if you want to reinstall the trap.
```

Portability

```
   ANSI
```

Returns

```
  If the trap can be established, the signal function returns a
  pointer to the previous handler function.  Otherwise, it returns
  the value SIG_ERR and places error information in the external
  integer  errno .
See Also

   raise
```

1.244 sin()

```
sin-Sine function

Synopsis

  #include <math.h>

  r = sin(x);

  double r;   /* result */
  double x;   /* angle */

Description

  The sin routine computes the sine of an angle expressed in
  radians.

Portability

   ANSI

See Also

   cos ,  cosh ,  __matherr ,  sinh
```

1.245 sinh()

```
sinh-Hyperbolic sine function

Synopsis

  #include <math.h>

  r = sinh(x);

  double r;   /* result */
  double x;   /* angle */

Description

  The sinh routine computes the normal hyperbolic sine of an angle.
  A range error occurs if the hyperbolic sine is too large to be
  represented by a double-precision floating-point number.
```

```
Portability

   ANSI

See Also

   cos ,  cosh ,  __matherr ,  sin
```

1.246 sizmem()

```
sizmem-Get memory pool size

Synopsis

  #include <stdlib.h>

  size = sizmem(void);

  long size;

Description

  This function returns the number of unallocated bytes in the
  current memory pool.  This value is the sum of the sizes of all
  unallocated blocks, so it does not indicate the size of the
  largest free block.

  Also, the value does not indicate the maximum amount of memory
  that can be allocated.  That is, the allocation functions will
  automatically expand the pool when no block of sufficient size is
  found in the pool.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

See Also

   getmem ,  getml ,  rlsmem ,  rlsml ,  rstmem
```

1.247 sprintf()

```
sprintf-Formatted print to a string

Synopsis

  #include <stdio.h>
  length = sprintf(s,fmt,arg1,arg2,...);

  int length;       /* number of characters generated */
  char *s;        /* pointer to a character string */
  const char *fmt;      /* format string */
  type argx;          /* arguments */
```

Description

  This function produces an output stream of ASCII characters and
  sends the output to the storage area whose address is given by the
  argument s. Make sure that this area is large enough to hold the
  maximum number of characters that might be generated. The sprintf
  function also generates a NULL byte to terminate the stored
  string.

  This function works like the fprintf function.   See the
  description of the fprintf function earlier in this chapter for a
  complete description.

Portability

   ANSI

Returns

  This function returns the number of output characters generated.
  This number does not include the terminating NULL byte.

Example

```
  /*
   * This example prints a message indicating whether
   * the function argument is positive or negative.
   * In the second printf, the width and precision
   * are 15 and 8, respectively.
   */
  #include <stdio.h>

  void pneg(double value)
  {
      char *sign, buff[256];

      if (value < 0) sign = "negative";
      else sign = "not negative";

      sprintf(buff,"The number %E is %s.\n", value, sign);
      printf(buff);
      sprintf(buff,"The number %*.*E is %s.\n", 15, 8, value, sign);
      printf(buff);
  }

  void main(void)
  {
      pneg(37.8);
      pneg(-18.2);
  }
```

See Also

```
   fprintf ,  printf
```

1.248 sqsort()

sqsort-Sort an array of short integers

Synopsis

```
  #include <stdlib.h>

  void sqsort(sa,n);

  short *sa;      /* pointer to short int array */
  size_t n;       /* number of elements in array */
```

Description

  The sqsort function sorts the specified array of short integers
  using the ACM 271 algorithm, more popularly known as Quicksort.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   SAS/C

See Also

```
   dqsort ,  fqsort ,  lqsort ,  qsort ,  tqsort
```

1.249 sqrt()

sqrt-Square root function

Synopsis

```
  #include <math.h>

  r = sqrt(x);

  double r, x;
```

Description

  The sqrt function calculates the square root of a number. The
  number must be a positive argument. If you supply a negative
  argument, the __matherr function will be called with a DOMAIN
  error.

```
Portability

   ANSI

See Also

   __matherr ,  pow ,  pow2
```

1.250 srand()

```
srand-Set the seed for the rand function

Synopsis

  #include <stdlib.h>

  void srand(seed);

  unsigned int seed;  /* random number seed */

Description

  The srand function resets the random number generator to a new
  seed value.  The initial default seed is 1.

Portability

   ANSI

Example

  /* This example prints 1000 random numbers */
  #include <stdio.h>
  #include <stdlib.h>

  void main(int argc, char *argv[])
  {
      int i, x;

      if (argc > 1)
      {
    stcd_i(argv[1],&x);
    if (x == 0)
    {
        x = 1;
    }
    printf("Seed value is %d\n",x);
    srand(x);
      }
      printf("Here are 1000 random numbers...\n");
      for (i = 0; i < 200; i++)
      {
    printf("%5d %5d %5d %5d %5d\n",
           rand(),rand(),rand(),rand(),rand());
      }
      printf("\n");
  }
```

See Also

   drand48 , erand48 , jrand48 , lrand48 ,
   nrand48 , rand , srand48

1.251 srand48()

srand48-Set high 32 bits of an internal seed

Synopsis

```
  #include <math.h>

  void srand48(hseed);
  long hseed;    /* high 32 bits of seed value */
```

Description

The srand48 function allows you to initialize the internal 48-bit seed used by the functions drand48 , erand48 , lrand48 , nrand48 , and jrand48 . The value you specify is copied into the high 32 bits of the seed, and the low 16 bits are set to 0x330E.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

   UNIX

See Also

   erand48 , lcong48 , rand , seed48 , srand

1.252 sscanf()

sscanf-Formatted input conversions

Synopsis

```
  #include <stdio.h>

  n = sscanf(ss,fmt,arg1,arg2,...);

  int n;       /*  number of input items matched, or EOF */
  const char *ss;     /*  input string (sscanf only) */
  const char *fmt;    /*  format string */
  ---- *argx;     /*  pointers to input data areas */
```

Description

  This function performs formatted input conversions on text
  obtained from a string.

  This function works like the fscanf function. See the description
  of the fscanf function earlier in this chapter for a complete
  description.

Portability

   ANSI

Returns

  This function returns the number of assignments that were made.
  For example, a return value of 3 indicates that conversion results
  were assigned to the arguments arg1, arg2, and arg3.

See Also

   fscanf , scanf

1.253 stackavail()

stackavail-Get current available stack size

Synopsis

```
  #include <dos.h>

  size = stackavail(void);
  unsigned long size;
```

Description

  This function calculates the amount of stack that is currently
  available. This function will not work in a shared library and is
  not reliable if you compile with the stackext option.

Portability

   SAS/C

Returns

  This function returns the number of bytes that are currently
  available on the stack.

See Also

   stacksize , stackused , __stack

1.254 stacksize()

```
stacksize-Get the current stack size

Synopsis


  #include <dos.h>

  size = stacksize(void);
  unsigned long size;      /* size of current stack */

Description

  This function calculates the size of the current stack and returns
  its size in bytes.  This function will not work in a shared
  library and is not reliable if you compile with the stackext
  option.

Portability

   SAS/C

Returns

  This function returns the number of bytes available for the entire
  stack when the program was started.

Example

  /*
   * Ensure at least 8K is available for certain
   * operations
   */
  #include <stdio.h>
  #include <dos.h>

  void main(void)
  {
      char *amount;

      if (stacksize() > 8000)
      {
    /* Do the operation */
    amount = "More than 8000 bytes";
      }
      else
      {
    /* Tell users it is not safe to do it */
    amount = "8000 bytes or less";
      }
      printf("%s stack available.\n", amount);
  }

See Also
```

stackavail , stackused , __stack

1.255 stackused()

stackused-Get the amount of stack in use

Synopsis

```
  #include <dos.h>

  size = stackused(void);

  unsigned long size;      /* amount of current stack in use */
```

Description

This function calculates the amount of the current stack currently
in use and returns the amount in bytes. This function will not
work in a shared library and is not reliable if you compile with
the stackext option.

Portability

SAS/C

Returns

This function returns the number of bytes of stack currently being
used.

See Also

stackavail , stacksize , __stack

1.256 stat()

stat-Get information on a file

Synopsis

```
  #include <sys/stat.h>

  rc = stat(file, st);

  int rc;     /* return code  */
  const char *file; /* file name */
  struct stat *st;  /* stat info structure */
```

Description

This function obtains information for the given file. If the file is not in the current directory, the file path must be included as part of the filename. Permission to read, write, or execute the file is not required.

This function is provided for compatibility with UNIX. For code that will be used only on the Amiga, use the AmigaDOS function Examine instead.

The information is placed into the stat structure pointed to by the st argument. The structure is defined in the file stat.h.

st is a pointer to a stat structure that must be allocated on a 4-byte (long word) boundary by the calling program. A common error is failing to allocate the structure before calling the function. You can make sure the structure is long-word aligned by either declaring it with the __aligned keyword or by allocating it dynamically with any SAS/C or system allocation function (such as the malloc or AllocMem function).

The following table lists defines that are combined with the logical OR operator to form the st_mode field. This list is found in sys/commifmt.h.

Symbol	Meaning
S_ISCRIPT	The object has its script protection bit set.
S_IPURE	The object is executable.
S_IARCHIVE	The file has its archive bit set.
S_IREAD	The file is readable.
S_IWRITE	The file is writable.
S_IEXECUTE	The file is executable.
S_IDELETE	The file is deletable.

Portability

UNIX

Returns

If the operation is successful, the function returns 0. Otherwise, it returns -1 and places error information in the external integers errno and _OSERR .

See Also

chmod , errno , fstat , _OSERR

1.257 stcarg()

stcarg-Get an argument

Synopsis

```
#include <string.h>
```

```
  length = stcarg(s,b);

  int length;    /* number of bytes in argument */
  const char *s;    /* text string pointer */
  const char *b;    /* break string pointer */
```

Description

  This function scans the text string until it finds one of the
  break characters or the NULL terminating byte. While scanning,
  the stcarg function skips over substrings that are enclosed in
  single or double quotes, and the backslash is recognized as an
  escape character. Break characters will not be detected if they
  are quoted or preceded by a backslash.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   SAS/C

Returns

  The function returns a count of the number of characters in the
  argument s up to but not including the break character or NULL
  terminator.

Example

```
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>

  void main(void)
  {
      char a[256],b[256];
      int x;

      while(1)
      {
    printf("Enter text string:  ");
    if (fgets(a,sizeof(a),stdin) == NULL)
    {
        break;
    }
    printf("Enter break string: ");
    if (fgets(b,sizeof(b),stdin) == NULL)
    {
        break;
    }
    x = stcarg(a,b);
    printf("Arg length: %d, Arg text: *s\n", x, a);
      }
      printf("\n");
  }
```

See Also

stpbrk , strcspn , strpbrk

1.258 stccpy()

stccpy-Copy one string to another

Synopsis

```
#include <string.h>

size = stccpy(to,from,n);

int size;      /* number of bytes copied */
char *to;      /* destination pointer */
const char *from;   /* source pointer */
int n;         /* maximum source length */
```

Description

This function copies the NULL-terminated source string to the destination area. The stccpy function writes up to n characters to the destination and stops copying at the first occurrence of a NULL byte. The stccpy function always produces a NULL-terminated string.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

UNIX

Returns

This function returns the actual number of bytes placed in the 'to' area, including the NULL terminator.

Example

```
/*
 * This example prints: Hello, my name is Flo.
 */
#include <stdio.h>
#include <string.h>

void main(void)
{
    char b[256];

    stccpy(b,"Hello, ",256);
    stccpy(&b[strlen(b)],"my name is ",256-strlen(b));
    stccpy(&b[strlen(b)],"Flo.",256-strlen(b));
```

```
      puts(b);
  }
```

1.259 stcd_i()

stcd_i-Convert a decimal string to an integer

Synopsis

```
  #include <string.h>

  length = stcd_i(in,ivalue);

  int length;      /* input length */
  const char *in;      /* input string pointer */
  int *ivalue;      /* integer value pointer */
```

Description

  This function scans an input string and converts the leading
  characters into short integers. The input string must begin with a
  plus sign (+), minus sign (-), or a decimal digit (0 to 9). The
  stcd_i function stops scanning the input string when it reaches
  the first invalid character. At that point, the resulting value is
  stored in the area addressed by the second argument.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   SAS/C

Returns

  This function returns the number of input characters converted.
  The result is 0 if the first character of the input string is not
  a valid decimal digit, a plus sign (+), or a minus sign (-). In
  that case, the conversion result pointed to by the second argument
  is 0.

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int x;
      int j;
      char b[80];

      while(1)
      {
    printf("\nEnter a decimal value: ");
    if (fgets(b,sizeof(b),stdin) == NULL)
    {
        break;
    }
    x = stcd_i(b,&j);
    printf("stcd_i: Length %d, Result %d\n",x,j);
      }
      printf("\n");
  }
```

1.260 stcd_l()

```
stcd_l-Convert a decimal string to a long integer

Synopsis

  #include <string.h>

  length = stcd_l(in,lvalue);

  int length;   /* input length */
  const char *in;   /* input string pointer */
  long *lvalue;   /* long integer value pointer */

Description

  This function scans an input string and converts the leading
  characters into long integers.  The input string must begin with a
  plus sign (+), minus sign (-), or a decimal digit (0 to 9).  The
  stcd_l function stops scanning the input string when it reaches
  the first invalid character.  At that point, the resulting value
  is stored in the area addressed by the second argument.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

Returns

  This function returns the number of input characters converted.
  The result is 0 if the first character of the input string is not
  a valid decimal digit, a plus sign (+), or a minus sign (-).  In
  that case, the conversion result pointed to by the second argument
  is also 0.

Example

  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int x;
      long j;
      char b[80];

      while(1)
      {
    printf("\nEnter a decimal value: ");
    if (fgets(b,sizeof(b),stdin) == NULL)
    {
        break;
    }
    x = stcd_l(b,&j);
    printf("stcd_l: Length %d, Result %ld\n",x,j);
      }
      printf("\n");
  }
```

1.261 stcgfe()

```
stcgfe-Get the filename extension

Synopsis

  #include <string.h>

  size = stcgfe(ext,name);

  int size;    /* size of result string */
  char *ext;     /* extension area pointer */
  const char *name;  /* file name pointer */

Description

  This function isolates the extension portion of a filename from
  the path and node.  The node is the rightmost portion of the
  filename that is separated from the rest of the name by a colon or
  a slash. The extension is the final part of the node that begins
  with a period, and the path is the leading part of the name up to
  the node.

  The maximum number of bytes copied into your array is defined in
  the file dos.h as FESIZE.  You should provide a buffer at least
  FESIZE bytes long.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

Returns

  The size value is the same as would be returned by the  strlen
  function.  That is, if size is 0, then the desired portion of the
  filename could not be found, and the result area contains a NULL
  string.
```

Example

```
  #include <stdio.h>
  #include <string.h>
  #include <dos.h>

  void main(void)
  {
      char file[256],path[FMSIZE],node[FNSIZE],ext[F

      while(gets(file) != NULL)
      {
    stcgfe(ext,file);
    stcgfn(node,file);
    stcgfp(path,file);
    printf("PATH: \"%s\"\n" "NODE: \"%s\"\n"
           "EXT:  \"%s\"\n", path, node, ext);
      }
  }
```

See Also

stcgfn , stcgfp , strsfn

1.262 stcgfn()

stcgfn-Get a filename from a path

Synopsis

```
  #include <string.h>

  size = stcgfn(node,name);

  int size;    /* size of result string */
  char *node;    /* node area pointer */
  const char *name;  /* file name pointer */
```

Description

This function isolates the node portion of a filename from the path and extension. The node is the rightmost portion of the filename that is separated from the rest of the name by a colon or a slash. The extension is the final part of the node that begins with a period, and the path is the leading part of the name up to the node.

The maximum number of bytes copied into your array is defined in the file dos.h as FESIZE. You should provide a buffer at least FESIZE bytes long.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Returns

The size value is the same as would be returned by the strlen function. That is, if size is 0, then the desired portion of the filename could not be found, and the result area contains a NULL string.

See Also

stcgfe , stcgfp , strsfn

1.263 stcgfp()

stcgfp-Get the file path

Synopsis

```
  #include <string.h>

  size = stcgfp(path,name);

  int size;      /* size of result string */
  char *path;      /* path area pointer */
  const char *name;    /* file name pointer */
```

Description

This function isolates the path portion of a filename from the node and extension. The node is the rightmost portion of the filename that is separated from the rest of the name by a colon, slash, or backslash. The extension is the final part of the node that begins with a period, and the path is the leading part of the name up to the node. The following table contains examples of how you can isolate the parts of a filename using the stcgfe , stcgfn , and stcgfp functions.

The maximum number of bytes copied into your array is defined in the file dos.h as FESIZE. You should provide a buffer at least FESIZE bytes long.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Returns

```
  The size value is the same as would be returned by the  strlen
  function.  That is, if size is 0, then the desired portion of the
  filename could not be found, and the result area contains a NULL
  string.
```

See Also

```
   stcgfe ,  stcgfn ,  strsfn
```

1.264 stch_i()

stch_i-Convert a hexadecimal string to an integer

Synopsis

```
  #include <string.h>

  length = stch_i(in,ivalue);

  int length;   /* input length */
  const char *in;   /* input string pointer */
  int *ivalue;    /* integer value pointer */
```

Description

```
  This function scans an unsigned string of hexadecimal digits and
  converts the leading characters into short integers.  The string
  can contain digits from 0 to 9 and letters from A to F or a to f.
  The stch_i function stops scanning when it reaches the first
  invalid character.  At that point, the resulting value is stored
  in the area addressed by the second argument.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

```
   SAS/C
```

Returns

```
  This function returns the number of input characters converted.
  The result is 0 if the first character of the input string is not
  a valid hexadecimal digit In that case, the conversion result
  pointed to by the second argument is 0.
```

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int x;
      int j;
      char b[80];

      while(1)
      {
    printf("\nEnter a hexadecimal value: ");
    if (fgets(b,sizeof(b),stdin) == NULL)
    {
       break;
    }
    x = stch_i(b,&j);
    printf("stch_l: Length %d, Result %x\n",x,j);
      }
      printf("\n");
  }
```

1.265 stch_l()

```
stch_l-Convert a hexadecimal string to a long integer

Synopsis

  #include <string.h>

  length = stch_l(in,lvalue);

  int length;      /* input length */
  const char *in;      /* input string pointer */
  long *lvalue;      /* long integer value pointer */

Description

  This function scans an unsigned string of hexadecimal digits and
  converts the leading characters into long integers.  The string
  can contain digits from 0 to 9 and letters from A to F or a to f.
  The stch_l function stops scanning the input string when it
  reaches the first invalid character. At that point, the resulting
  value is stored in the area addressed by the second argument.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

Returns

  This function returns the number of input characters converted.
  The result is 0 if the first character of the input string is not
  a valid hexadecimal digit.  In that case, the conversion result
  pointed to by the second argument is 0.

Example

  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int x;
      long j;
      char b[80];

      while (1)
      {
    printf("\nEnter a hexadecimal value: ");
    if (fgets(b,sizeof(b),stdin) == NULL)
    {
        break;
    }
    x = stch_l(b,&j);
    printf("stch_l: Length %d, Result %lx\n",x,j);
      }
      printf("\n");
  }
```

1.266 stci_d()

```
stci_d-Convert an integer to a decimal string

Synopsis

  #include <string.h>

  length = stci_d(out,ivalue);

  int length;   /* output length */
  char *out;    /* output buffer pointer */
  int ivalue;   /* integer value */

Description

  This function converts an integer into an ASCII string that is the
  decimal equivalent of the integer.  The output area should be at
  least 7 bytes long, which is large enough to accommodate the
  maximum possible string, including the terminating NULL byte that
  this function appends.

  The first output character is a minus sign if the input value is
  negative.  No special leading character is generated if the value
  is positive.  Leading zeroes are suppressed, and a single 0
  character is generated if the input value is 0.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C
```

Returns

  The return value is the number of characters actually placed into
  the output area, not including the final NULL byte.

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int i,x;
      char b[7];

      while (1)
      {
    printf("\nEnter an integer: ");
    if (scanf("%d",&i) == EOF)
    {
        break;
    }
    x = stci_d(b,i);
    printf("stci_d: Length %d, Result %s\n",x,b);
      }
      printf("\n");
  }
```

See Also

   stci_h , stci_o , stcl_d , stcl_h , stcl_o , stcu_d , stcul_d

1.267 stci_h()

stci_h-Convert an integer to a hexadecimal string

Synopsis

```
  #include <string.h>

  length = stci_h(out,ivalue);

  int length;    /* output length */
  char *out;     /* output buffer pointer */
  int ivalue;    /* integer value */
```

Description

  This function converts an integer into an ASCII string that is the
  hexadecimal equivalent of the integer. The output area should be
  at least 5 bytes long, which is large enough to accommodate the
  maximum possible string, including the terminating NULL byte that
  this function appends.

  Leading zeroes are suppressed, and a single 0 character is

```
  generated if the input value is 0.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

```
   SAS/C
```

Returns

```
  The return value is the number of characters actually placed into
  the output area, not including the final NULL byte.
```

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int i,x;
      char b[5];

      while (1)
      {
    printf("\nEnter an integer: ");
    if (scanf("%d",&i) == EOF)
    {
        break;
    }
    x = stci_h(b,i);
    printf("stci_h: Length %d, Result %s\n",x,b);
      }
      printf("\n");
  }
```

See Also

```
   stci_d ,  stci_o ,  stcl_d ,  stcl_h ,  stcl_o ,  stcu_d ,  stcul_d
```

1.268 stci_o()

stci_o-Convert an integer to an octal string

Synopsis

```
  #include <string.h>

  length = stci_o(out,ivalue);

  int length;  /* output length */
  char *out;   /* output buffer pointer */
  int ivalue;  /* integer value */
```

```
Description

  This function converts an integer into an ASCII string that is the
  octal equivalent of the integer.  The output area should be at
  least 7 bytes long, which is large enough to accommodate the
  maximum possible string, including the terminating NULL byte that
  this function appends.

  Leading zeroes are suppressed, and a single 0 character is
  generated if the input value is 0.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

Returns

  The return value is the number of characters actually placed into
  the output area, not including the final NULL byte.

Example

  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int i,x;
      char b[7];

      while (1)
      {
    printf("\nEnter an integer: ");
    if (scanf("%d",&i) == EOF)
    {
        break;
    }
    x = stci_o(b,i);
    printf("stci_o: Length %d, Result %s\n",x,b);
      }
      printf("\n");
  }

See Also

   stci_d ,  stci_h ,  stcl_d ,  stcl_h ,  stcl_o ,  stcu_d ,  stcul_d
```

1.269 stcis()

```
stcis-Count the number of string characters in the set

Synopsis
```

```
  #include <string.h>

  length = stcis(s,b);

  int length;   /* span length in bytes */
  const char *s;    /* points to string being scanned */
  const char *b;    /* points to character set string */
```

```
Description

  This function counts the number of characters at the beginning of
  the string s that are included in the character set specified by
  the argument b. For example, if string s is hello, and set b
  includes the characters h, e, and l, the stcis function returns a
  4.  The stcis function is provided for compatibility with previous
  versions of the compiler.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

Returns

  This function returns the number of bytes that are in the
  specified character set.  The scan always stops when the NULL
  terminator byte is reached.

Example

  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      char s1[256],s2[256];

      while(1)
      {
    printf("\nEnter test string: ");
    if (fgets(s1,sizeof(s1),stdin) == NULL)
    {
        break;
    }
    printf("Enter span string: ");
    if (fgets(s2,sizeof(s2),stdin) == NULL)
    {
        break;
    }
    printf("stcis: %d\n",stcis(s1,s2));
      }
      printf("\n");
  }
```

See Also

   stcisn , strspn

1.270 stcisn()

stcisn-Count the number of string characters
not in the set

Synopsis

```
  #include <string.h>

  length = stcisn(s,b);

  int length;  /* span length in bytes */
  const char *s;   /* points to string being scanned */
  const char *b;   /* points to character set string */
```

Description

  This function counts the number of characters at the beginning of
  the string s that are not included in the character set specified
  by the argument b. For example, if string s is hello, and set b
  includes the characters h, e, and l, the stcisn function returns a
  0. The stcisn function is provided for compatibility with previous
  versions of the compiler.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   SAS/C

Returns

  This function returns the number of bytes that are not in the
  specified character set. The scan always stops when the NULL
  terminator byte is reached.

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      char s1[256],s2[256];

      while(1)
      {
    printf("\nEnter test string: ");
    if (fgets(s1,sizeof(s1),stdin) == NULL)
    {
        break;
    }
    printf("Enter span string: ");
    if (fgets(s2,sizeof(s2),stdin) == NULL)
    {
        break;
    }
    printf("stcisn: %d\n",stcisn(s1,s2));
      }
      printf("\n");
  }
```

See Also

```
   stcis ,  strcspn
```

1.271 stcl_d()

stcl_d-Convert a long integer to a decimal string

Synopsis

```
  #include <string.h>

  length = stcl_d(out,lvalue);

  int length; /* output length */
  char *out;  /* output buffer pointer */
  long lvalue;  /* long integer value */
```

Description

This function converts a long integer into an ASCII string that is the decimal equivalent of the integer. The output area should be at least 13 bytes long, which is large enough to accommodate the maximum possible string, including the terminating NULL byte that this function appends.

The first output character is a minus sign if the input value is negative. No special leading character is generated if the value is positive. Leading zeroes are suppressed, and a single 0 character is generated if the input value is 0.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Returns

The return value is the number of characters actually placed into the output area, not including the final NULL byte.

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int x;
      long l;
      char b[13];

      while (1)
      {
    printf("\nEnter an integer: ");
    if (scanf("%ld",&l) == EOF)
    {
        break;
    }
    x = stcl_d(b,l);
    printf("stcl_d: Length %d, Result %s\n",x,b);
      }
      printf("\n");
  }
```

See Also

stci_d , stci_h , stci_o , stcl_h , stcl_o , stcu_d , stcul_d

1.272 stcl_h()

stcl_h-Convert a long integer to a hexadecimal string

Synopsis

```
  #include <string.h>

  length = stcl_h(out,lvalue);

  int length; /* output length */
  char *out;  /* output buffer pointer */
  long lvalue;  /* long integer value */
```

Description

This function converts a long integer into an ASCII string that is the hexadecimal equivalent of the integer. The output area should be at least 9 bytes long, which is large enough to accommodate the maximum possible string, including the terminating NULL byte that each function appends.

Leading zeroes are suppressed, and a single 0 character is generated if the input value is 0.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

   SAS/C

Returns

  The return value is the number of characters actually placed into
  the output area, not including the final NULL byte.

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int x;
      long l;
      char b[9];

      while (1)
      {
    printf("\nEnter an integer: ");
    if (scanf("%ld",&l) == EOF)
    {
        break;
    }
    x = stcl_h(b,l);
    printf("stcl_h: Length %d, Result %s\n",x,b);
      }
      printf("\n");
  }
```

See Also

   stci_d , stci_h , stci_o , stcl_d , stcl_o , stcu_d , stcul_d

1.273 stcl_o()

stcl_o-Convert a long integer to an octal string

Synopsis

```
  #include <string.h>

  length = stcl_o(out,lvalue);

  int length; /* output length */
  char *out;  /* output buffer pointer */
  long lvalue;  /* long integer value */
```

Description

  This function converts a long integer into an ASCII string that is

```
  the octal equivalent of the integer.  The length of the output
  area should be at least 12 bytes long, which is large enough to
  accommodate the maximum possible string, including the terminating
  NULL byte that this function appends.

  Leading zeroes are suppressed, and a single 0 character is
  generated if the input value is 0.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

```
   SAS/C
```

Returns

```
  The return value is the number of characters actually placed into
  the output area, not including the final NULL byte.
```

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int x;
      long l;
      char b[13];

      while (1)
      {
    printf("\nEnter an integer: ");
    if (scanf("%ld",&l) == EOF)
    {
        break;
    }
    x = stcl_o(b,l);
    printf("stcl_o: Length %d, Result %s\n",x,b);
      }
      printf("\n");
  }
```

See Also

```
   stci_d ,  stci_h ,  stci_o ,  stcl_d ,  stcl_h ,  stcu_d ,  stcul_d
```

1.274 stclen()

stclen-Measure the length of a string

Synopsis

```
  #include <string.h>
  length = stclen(s);

  size_t length;   /* number of bytes in s (before null) */
  const char *s;
```

Description

This function, which is equivalent to the ANSI Standard function strlen , returns the number of bytes in the string s before the NULL terminator byte.

This function is implemented as a macro and is provided only for compatibility with previous releases.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

OLD

Returns

The length equals the number of bytes in the string before the NULL byte.

See Also

strlen

1.275 stco_i()

stco_i-Convert an octal string to an integer

Synopsis

```
  #include <string.h>

  length = stco_i(in,ivalue);

  int length;   /* input length */
  const char *in;   /* input string pointer */
  int *ivalue;    /* integer value pointer */
```

Description

This function scans an unsigned octal string and converts the leading characters into short integers. The string can consist of octal digits 0 to 7. The stco_i function stops scanning the input string when it reaches the first invalid character. At that point, the resulting value is stored in the area addressed by the second argument.

This function is not available if the _STRICT_ANSI flag has been

```
  defined.

Portability

   SAS/C

Returns

  This function returns the number of input characters converted.
  The result is 0 if the first character of the input string is not
  a valid octal digit. In that case, the conversion result pointed
  to by the second argument is 0.

Example

  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int x;
      int j;
      char b[80];

      while(1)
      {
    printf("\nEnter an octal value: ");
    if (fgets(b,sizeof(b),stdin) == NULL)
    {
        break;
    }
    x = stco_i(b,&j);
    printf("stco_i: Length %d, Result %x\n",x,j);
      }
      printf("\n");
  }

See Also

   stcd_i ,  stcd_l ,  stch_i ,  stch_l ,  stco_l
```

1.276 stco_l()

```
stco_l-Convert an octal string to a long integer

Synopsis

  #include <string.h>

  length = stco_l(in,lvalue);

  int length;     /* input length */
  const char *in;     /* input string pointer */
  long *lvalue;     /* long integer value pointer */
```

```
Description

  This function scans an unsigned octal string and converts the
  leading characters into long integers.  The string can consist of
  octal digits 0 to 7. The stco_l function stops scanning the input
  string when it reaches the first invalid character. At that point,
  the resulting value is stored in the area addressed by the second
  argument.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

Returns

  This function returns the number of input characters converted.
  The result is 0 if the first character of the input string is not
  a valid octal digit.  In that case, the conversion result pointed
  to by the second argument is 0.

Example

  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      int x;
      long j;
      char b[80];

      while(1)
      {
    printf("\nEnter an octal value: ");
    if (fgets(b,sizeof(b),stdin) == NULL)
    {
        break;
    }
    x = stco_l(b,&j);
    printf("stco_l: Length %d, Result %lx\n",x,j);
      }
      printf("\n");
  }

See Also

   stcd_i ,  stcd_l ,  stch_i ,  stch_l ,  stco_i
```

1.277 stcpm()

```
stcpm-Unanchored pattern match
```

```
Synopsis

  #include <string.h>

  size = stcpm(string,pattern,match);

  int size;   /* size of matching string */
  const char *string; /* string to be scanned */
  const char *pattern;  /* pattern string */
  char **match;   /* returns pointer to matching string */

Description

  This function scans a string to find a specified pattern.  You can
  use the following wildcard characters to specify a pattern:

      Pattern Matches
      ------- -------
         ?  any single character
         c* zero or more occurrences of character c
         c+ one or more occurrences of character c
         \? a question mark (?)
         \* an asterisk (*)
         \+ a plus sign (+)

  Any other character must match exactly.  The following table lists
  some examples.

      Pattern Matches
      ------- -------
        abc only abc
        ab*c  ac, abc, or abbc
        ab+c  abc, abbc, or abbbc
        ab?*c a string starting with ab and ending in c, for
      example, abxyzc
        ab\*c only ab*c

  The match can occur anywhere in the string.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

Returns

  The function returns the size of the matching string or 0 if there
  was no match.  It also returns a pointer to the beginning of the
  matching string in the parameter match.

Example

  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  void main(void)
  {
      char s[100],p[100],*r;
      int x;

      while(1)
      {
    printf("\nSearch string => ");
    if (gets(s) == NULL)
    {
        break;
    }
    printf("Pattern       => ");
    if (gets(p) == NULL)
    {
        break;
    }
    x = stcpm(s,p,&r);
    if (x)
    {
        printf("stcpm: size = %d, match = \"%.*s\"\n", x, x, r);
    }
    else
    {
        printf("stcpm: no match\n");
        exit(EXIT_FAILURE);
    }
      }
      printf("\n");
  }
```

See Also

   astcsma , stcpma , stcsma

1.278 stcpma()

stcpma-Anchored pattern match

Synopsis

```
  #include <string.h>

  size = stcpma(string,pattern);

  int size;   /* size of matching string */
  const char *string; /* string to be scanned */
  const char *pattern;  /* pattern string */
```

Description

  This function scans a string to find a specified pattern. You can
  use the following wildcard characters to specify a pattern:

```
      Pattern Matches
      ------- -------
         ?  any single character
         c* zero or more occurrences of character c
         c+ one or more occurrences of character c
         \? a question mark (?)
         \* an asterisk (*)
         \+ a plus sign (+)
```

Any other character must match exactly. The following table lists some examples.

```
      Pattern Matches
      ------- -------
        abc only abc
        ab*c  ac, abc, or abbc
        ab+c  abc, abbc, or abbbc
        ab?*c a string starting with ab and ending in c, for
      example, abxyzc
        ab\*c only ab*c
```

The match must occur at the beginning of the string.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Returns

The function returns the size of the matching string or 0 if there was no match.

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      char s[100],p[100];
      int x;

      while(1)
      {
    printf("\nSearch string => ");
    if (gets(s) == NULL)
    {
        break;
    }
    printf("Pattern       => ");
    if (gets(p) == NULL)
    {
        break;
    }
    x = stcpma(s,p);
    if (x)
    {
        printf("stcpma: size = %d, match = \"%.*s\"\n", x,x,s);
    }
    else
    {
        printf("stcpma: no match\n"):
    }
      }
      printf("\n");
  }
```

See Also

   astcsma , stcpm , stcsma

1.279 stcsma()

stcsma-UNIX string pattern match (anchored)

Synopsis

```
  #include <string.h>

  length = stcsma (s,p);

  int length;    /* length of matching string */
  const char *s;     /* string being scanned */
  const char *p;     /* pattern string */
```

Description

The function stcsma performs an anchored pattern match of the type used by the UNIX shell. The only meta-characters recognized are the asterisk (*) and the question mark (?). The asterisk matches an arbitrary number of characters, and the question mark matches exactly one character. The pattern must match at the beginning of the supplied string.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

   AmigaDOS

Returns

This function returns the length of the matching string or 0 if there was no match.

See Also

```
   astcsma ,  stcpm ,  stcpma
```

1.280 stcu_d()

```
stcu_d-Convert an unsigned integer to a decimal string

Synopsis

  #include <string.h>

  length = stcu_d(out,uivalue);

  int length;     /* output length */
  char *out;      /* output buffer pointer */
  unsigned uivalue;  /* unsigned value */

Description

  This function converts an unsigned integer into an ASCII string
  that is the decimal equivalent of the integer.  The length of the
  output area should be at least 6 bytes long, which is large enough
  to accommodate the maximum possible string, including the
  terminating NULL byte that this function appends.

  Leading zeroes are suppressed, and a single 0 character is
  generated if the input value is 0.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

Returns

  The return value is the number of characters actually placed into
  the output area, not including the final NULL byte.

Example

  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      unsigned int i;
      int x;
      char b[13];

      while(1)
      {
    printf("\nEnter an integer: ");
    if (scanf("%u",&i) == EOF)
    {
        break;
    }
    x = stcu_d(b,i);
    printf("stcu_d: Length %d, Result %s\n",x,b);
      }
      printf("\n");
  }
```

See Also

stci_d , stci_h , stci_o , stcl_d , stcl_h , stcl_o , stcul_d

1.281 stcul_d()

stcul_d-Convert an unsigned long integer to a decimal string

Synopsis

```
#include <string.h>

length = stcul_d(out,ulvalue);

int length;    /* output length */
char *out;     /* output buffer pointer */
unsigned long ulvalue;   /* unsigned long integer value */
```

Description

This function converts an unsigned long integer into an ASCII string that is the decimal equivalent of the integer. The output area should be at least 12 bytes long, which is large enough to accommodate the maximum possible string, including the terminating NULL byte that each function appends.

Leading zeroes are suppressed, and a single 0 character is generated if the input value is 0.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Returns

The return value is the number of characters actually placed into the output area, not including the final NULL byte.

Example

```
#include <stdio.h>
#include <string.h>

void main(void)
```

```
  {
      int x;
      unsigned long i;
      char b[12];

      while(1)
      {
    printf("\nEnter an integer: ");
    if (scanf("%lu",&i) == EOF)
    {
        break;
    }
    x = stcul_d(b,i);
    printf("stcul_d: Length %d, Result %s\n",x,b);
      }
      printf("\n");
  }
```

See Also

stci_d , stci_h , stci_o , stcl_d , stcl_h , stcl_o

1.282 stpblk()

stpblk-Skip blanks

Synopsis

```
  #include <string.h>

  q = stpblk(p);

  char *q;    /* updated string pointer */
  const char *p;    /* string pointer */
```

Description

This function advances the string pointer past blank characters, that is, past all the characters for which the isspace function is true.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Returns

The function returns a pointer to the next nonblank character. The NULL terminator byte is not considered a blank, and so the function will not go past the end of the string.

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      char input[256];

      while(1)
      {
    puts("\nEnter a string with leading blanks...");
    if (gets(input) == NULL)
    {
        break;
    }
    printf("%s\n",stpblk(input));
      }
      printf("\n");
  }
```

See Also

   stcis , strspn

1.283 stpbrk()

stpbrk-Find a break character in a string

Synopsis

```
  #include <string.h>

  p = stpbrk(s,b);

  char *p;     /* points to break character in s */
  const char *s;     /* string to be scanned */
  const char *b;     /* break characters  */
```

Description

This function scans string s to find the first occurrence of a character from break string b.

This function is provided for compatibility with previous releases of the compiler. The ANSI function strpbrk is equivalent to the stpbrk function.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

   OLD

Returns

```
  If no character from string b is found in string s, a NULL pointer
  is returned.  Otherwise, p is a pointer to the first break
  character.
```

See Also

   strcspn , strpbrk , strspn

1.284 stpchr()

stpchr-Find a character in a string

Synopsis

```
  #include <string.h>

  p = stpchr(s,c);

  char *p;    /* updated string pointer */
  const char *s;    /* input string pointer */
  int c;      /* character to be located */
```

Description

```
  The stpchr function scans the input string to find the first
  occurrence of the character specified by argument c.  The stpchr
  function is provided for compatibility with previous releases of
  the compiler.  The ANSI function  strchr  is equivalent to the
  stpchr function.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

   OLD

Returns

```
  The stpchr function returns a NULL pointer if the input string is
  empty or if the specified character is not found.
```

See Also

   stpchrn , strchr , strrchr

1.285 stpchrn()

stpchrn-Find a character not in a string

Synopsis

```
  #include <string.h>

  p = stpchrn(s,c);

  char *p;    /* updated string pointer */
  const char *s;    /* input string pointer */
  int c;      /* character to be located */
```

Description

The stpchrn function scans the input string for the first occurrence of some character other than that specified in the c argument. This function is provided for compatibility with previous releases of the compiler. The ANSI function strrchr is equivalent to the stpchrn function.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

OLD

Returns

The stpchrn function returns a NULL pointer if the input string is empty or consists entirely of character c.

See Also

stpchr , strchr , strrchr

1.286 stpcpy()

stpcpy-Copy one string to another

Synopsis

```
  #include <string.h>

  np = stpcpy(to,from);

  char *np;     /* points to end of destination string */
  char *to;     /* destination pointer */
  const char *from;   /* source pointer */
```

Description

This function copies the NULL-terminated source string to the destination area. The entire source string is copied, and the resulting destination is always NULL-terminated.

This function is not available if the _STRICT_ANSI flag has been defined.

```
Portability

   SAS/C

Returns

  The stpcpy function returns a pointer to the end of the
  destination string, which is useful when you are building a string
  from several pieces.

  NOTE:  The ANSI  strcpy  function returns the 'to' string, but this
  function returns a pointer to the NULL byte after the 'to' string.

Example

  /*
   * This example should print: Hello, my name is Flo.
   */
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      char b[256],*p;

      p = stpcpy(b,"Hello, ");
      p = stpcpy(p,"my name is ");
      p = stpcpy(p,"Flo.");
      puts(b);
  }

See Also

   stccpy ,  strcpy ,  strncpy
```

1.287 stpdate()

```
stpdate-Convert a date array to a string

Synopsis

  #include <string.h>

  np = stpdate(p,mode,date);

  char *np;     /* updated output string pointer */
  char *p;      /* output string pointer       */
  int mode;     /* conversion mode          */
  const char *date;   /* date array, as follows      */
          /* date[0] => year - 1980      */
          /* date[1] => month (1 to 12)    */
          /* date[2] => day (1 to 31)      */

Description
```

```
  This function converts a 3-byte date array into ASCII or BCD
  according to the mode argument:

      Mode   Format    Type   Length
      ----   ------    ----   ------
        0    yymmdd    BCD    3 bytes
        1    yymmdd    ASCII  7 bytes
        2    mm/dd/yy    ASCII  9 bytes
        3    mm-dd-yy    ASCII  9 bytes
        4    MMM d yyyy    ASCII  up to 13 bytes
        5    Mm...m d, yyyy  ASCII  up to 19 bytes
        6    dd MMM yy     ASCII  10 bytes
        7    dd MMM yyyy     ASCII  12 bytes

  In the above formats, MMM represents a 3-character month
  abbreviation in capitals, and Mm...m represents the full month
  name (for example, January).  The mm, dd, and yy terms are
  2-character month, day, and year, respectively, while d is the
  date with the leading zero suppressed.  The yyyy term is the
  4-character year obtained by adding 1980 to the first byte of the
  date array.

  For all modes except 0, a NULL byte is appended to the output
  string.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

   SAS/C

Returns

```
  The function does not make validity checks on the date array.  It
  returns a pointer to the first byte past the generated output. For
  modes other than 0, this is a pointer to the NULL terminator byte.
```

See Also

   getclk , getft , stptime

1.288 stpsym()

stpsym-Get the next symbol from a string

Synopsis

```
  #include <string.h>

  p = stpsym(s,sym,symlen);

  char *p;       /* points to next input character */
  const char *s;       /* input string */
  char *sym;       /* output string */
  int symlen;      /* sizeof(sym) */
```

Description

This function breaks out the next symbol from the input string. The first character of the symbol must be alphabetic (upper- or lowercase), and the remaining characters must be alphanumeric. The pointer is not advanced past any initial white space in the input string.

The output string is the NULL-terminated symbol and will be an empty string if no symbol is found. If the symbol is longer than symlen-1, its excess characters are dropped.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Returns

The function returns a pointer to the next character past the symbol.

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      char a[256], b[10];
      char *p;

      while(1)
      {
    printf("\nEnter text string:  ");
    if (gets(a) == NULL)
    {
        break;
    }

    p = a;
    while(1)
    {
        p = stpsym(p,b,sizeof(b));
        printf("Symbol:   \"%s\"\n" "Residual: \"%s\"\n\n",b,p);
        if (b[0] == '\0')
        {
      break;
        }
        p++;
    }
      }
      printf("\n");
  }
```

See Also

   stcarg , stpbrk , strcspn , strpbrk

1.289 stptime()

stptime-Convert a time array to a string

Synopsis

```
  #include <string.h>

  np = stptime(p,mode,time);

  char *np;      /* updated output string pointer */
  char *p;       /* output string pointer */
  int mode;      /* conversion mode */
  const char *time;   /* time array, as follows */
          /* time[0] => hour (0 to 23) */
          /* time[1] => minute (0 to 59) */
          /* time[2] => second (0 to 59) */
          /* time[3] => hundredths (0 to 99) */
```

Description

  This function converts a 4-byte time array into ASCII or BCD
  according to the mode argument:

```
      Mode  Format       Type   Length
      ----  ------       ----   ------
        0   hhmmssdd     BCD    4 bytes
        1   hhmmss       ASCII  7 bytes
        2   hh:mm:ss     ASCII  9 bytes
        3   hhmmssdd     ASCII  9 bytes
        4   hh:mm:ss.dd  ASCII  12 bytes
        5   hh:mm        ASCII  6 bytes
        6   hr:mm:ss HH  ASCII  12 bytes
        7   hr:mm HH     ASCII  9 bytes
```

  The hh, mm, ss, and dd terms are the 2-digit (BCD or ASCII)
  equivalents of the binary values in the time array. The hr term is
  the 2-digit hour using the 12-hour form, and the HH term is either
  AM or PM.

  A NULL terminator byte is appended to the ASCII output strings.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   SAS/C

Returns

```
  The function does not make validity checks on the time array.  It
  returns a pointer to the first byte past the generated output. For
  modes other than 0, this is a pointer to the NULL terminator byte.
```

See Also

```
   getclk ,  getft ,  stpdate
```

1.290 stptok()

stptok-Get the next token from a string

Synopsis

```
  #include <string.h>

  p = stptok(s,tok,toklen,brk);

  char *p;       /* points to next character after token */
  const char *s;       /* points to input string */
  char *tok;       /* points to output buffer */
  int toklen;      /* size of buffer pointed to by tok */
  const char *brk;     /* break string */
```

Description

```
  This function breaks out the next token from the input string and
  moves it to the token buffer with a NULL terminator.  A token
  consists of all characters in the input string s up to but not
  including the first character that is in the break string.  In
  other words, the brk argument specifies the characters that cannot
  be included in a token.

  If the input string begins with a break character, then the token
  buffer will contain a NULL string, and the return pointer p will
  be the same as the argument s.  If no break character is found
  after toklen-1 input characters have been moved to the token
  buffer, or if the input string terminator (a NULL byte) is
  reached, then the scan stops as if a break character were reached.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

```
   SAS/C
```

Returns

```
  The function returns a pointer to the next character in the input
  string.
```

Example

```
/*
 * This example breaks out words that are
 * separated by blanks or commas.
 * The token buffer takes on the following
 * values as the program loops:
 *
 *   LOOP   TOKEN
 *    1     first
 *    2     second
 *    3     third
 *    4     fourth
 */
#include <stdio.h>
#include <string.h>

char test[] = "first, second third, fourth";

void main(void)
{
    char *p = test;
    char token[50];

    while(1)
    {
  p = stptok(p,token,sizeof(token)," ,");
  printf("%s\n",token);
  if (*p == '\0')
  {
      break;
  }
  p = stpblk(++p);
    }
}
```

See Also

stpblk , strtok

1.291 strbpl()

strbpl-Build a string pointer list

Synopsis

```
#include <string.h>

n = strbpl(s,max,t);

int n;        /* number of pointers */
char *s[];      /* pointer to string pointer list */
int max;      /* maximum number of pointers */
const char *t;      /* text pointer */
```

Description

```
  This function constructs a list of pointers to the strings
  contained within the specified text array.  Each string must be
  NULL-terminated, and the text array must be terminated by a NULL
  string.  In other words, array t must end with two NULL bytes, one
  to terminate the final string and another to terminate the array.
  The string pointer list s is terminated by a NULL pointer.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

```
   SAS/C
```

Returns

```
  The return value indicates how many string pointers were placed
  into array s.  If the number of strings plus the final NULL
  pointer is greater than the argument max, a value of -1 is
  returned.
```

Example

```
  #include <stdio.h>
  #include <string.h>

  char text[] = "string 1\0string 2\0";

  void main(void)
  {
      char *list[5];
      int count, i;

      /*
       * The following call has the following effect:
       *
       *    Return value (count) is 2.
       *    list[0] => "string 1"
       *    list[1] => "string 2"
       *    list[2] => NULL
       *
       */
      count = strbpl(list,5,text);
      printf("%d strings were found:\n", count);
      for (i=0; i<count; i++)
      {
    printf("%s\n", list[i]);
      }
  }
```

See Also

```
   getfnl ,  strsrt
```

1.292 strcat()

```
strcat-Concatenate strings

Synopsis

  #include <string.h>

  p = strcat(to,from);

  char *p;       /* same as destination string pointer */
  char *to;      /* destination string pointer */
  const char *from;    /* source string pointer */

Description

  This function concatenates the source string to the end of the
  destination string, overwriting the existing NULL byte at the end
  of the destination string.  The strcat function places a NULL byte
  at the new end of the destination string.

Portability

   ANSI

Returns

  This function returns a pointer that is the same as the first
  argument.

Example

  #include <stdio.h>
  #include <string.h>

  char first[100];
  char *second=" a test";

  void main(void)
  {
      strcpy(first, "This is");
      strcat(first, second);
      printf("%s\n", first);
      /* output is "This is a test" */
  }

See Also

   stpcpy ,  strncat
```

1.293 strchr()

```
strchr-Find a character in a string
```

Synopsis

```
  #include <string.h>

  p = strchr(s,c);

  char *p;     /* updated string pointer */
  const char *s;     /* input string pointer */
  int c;       /* character to be located */
```

Description

  The strchr function scans the input string to find the first
  occurrence of the character specified by the argument c.

Portability

   ANSI

Returns

  A NULL pointer is returned if the input string is empty or if the
  specified character is not found. Otherwise, this function
  returns a pointer to the first matching character in the argument
  s.

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      char *p;

      p=strchr("This is a test",'t');

      /* p now points to "test" */
      printf("%s\n",p);

      p=strchr("This is a test",'s');

      /* p now points to "s is a test" */
      printf("%s\n",p);
  }
```

See Also

   stpchr , stpchrn , strrchr

1.294 strcmp()

strcmp-Compare strings, case sensitive

Synopsis

```
  #include <string.h>

  x = strcmp(a,b);

  int x;          /* comparison result */
  const char *a,*b;     /* strings being compared */
```

Description

```
  This function compares two NULL-terminated strings.  The ASCII
  collating sequence is used in all cases.

  The relative collating sequence of the strings is indicated by the
  sign of the return value, as follows:

      Sign      Meaning
      ----      -------
      negative  first string is below the second
      zero      strings are equal
      positive  first string is above the second

  If the strings have different lengths, the shorter one is treated
  as if it were extended with zeroes.  The strcmp function has a
  built-in version that is equivalent to the standard library
  version.  The statement #include <string.h> provides a default
  setting by which built-in functions are accessed.  If you do not
  want the built-in function, you can enter an #undef strcmp after
  including the string.h file.
```

Portability

```
   ANSI
```

Returns

```
  The sign of the return value indicates the relative collating
  sequence of the strings, as indicated above.
```

Example

```
  #include <stdio.h>
  #include <string.h>

  void result(char *name, int r)
  {
      char *p;

      if (r == 0)
      {
    p = "is equal to";
      }
      if (r < 0)
      {
    p = "is less than";
      }
      if (r > 0)
      {
    p = "is greater than";
      }
      printf("%s string A %s string B\n",name,p);
  }

  void main(void)
  {
      char a[256],b[256];

      while(1)
      {
    printf("Enter string A: ");
    if (fgets(a,sizeof(a),stdin) == NULL)
    {
        break;
    }
    printf("Enter string B: ");
    if (fgets(b,sizeof(b),stdin) == NULL)
    {
        break;
    }
    result("strcmp: ",strcmp(a,b));
      }
      printf("\n");
  }
```

See Also

strcmpi , stricmp , strncmp , strnicmp

1.295 strcmpi()

strcmpi-Compare strings, case insensitive

Synopsis

```
  #include <string.h>

  x = strcmpi(a,b);

  int x;          /* comparison result */
  const char *a,*b;     /* strings being compared */
```

Description

This function compares two NULL-terminated strings using the ASCII collating sequence. The strcmpi function does not distinguish between uppercase and lowercase. This function is a hold-over from various Microsoft compilers. Use the stricmp function in new code.

The relative collating sequence of the strings is indicated by the sign of the return value, as follows:

```
     Sign      Meaning
     ----      -------
     negative  first string is below the second
     zero      strings are equal
     positive  first string is above the second
```

If the strings have different lengths, the shorter one is treated as if it were extended with zeroes.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

OLD

Returns

The sign of the return value indicates the relative collating sequence of the strings, as indicated above.

See Also

strcmp , stricmp , strncmp , strnicmp

1.296 strcoll()

strcoll-Compare strings based on locale

Synopsis

```
  #include <string.h>

  res = strcoll(s1, s2);

  int res;      /* result of comparison */
  const char *s1,*s2;   /* strings to compare   */
```

Description

This function compares two strings based on the current locale. Because of the limitations of the ANSI specifications, this function cannot correctly handle the rules of the German collating sequence completely.

Portability

ANSI

Returns

This function returns a 0 if both strings are equal. If string s1 is logically less than string s2, the return value is less than 0. If string s1 is logically greater than string s2, the return value is greater than 0.

Example

```
  if (!strcoll("string1", "string2"))
      printf("This is funny, they shouldn’t match\n");
```

See Also

   setlocale

1.297 strcpy()

strcpy-Copy one string to another

Synopsis

```
  #include <string.h>

  p = strcpy(to,from);

  char *p;      /* same as destination pointer */
  char *to;     /* destination pointer */
  const char *from;   /* source pointer */
```

Description

  This function copies the entire NULL-terminated source string to
  the destination area. The resulting destination is always
  NULL-terminated.

  The strcpy function has a built-in version that is equivalent to
  the standard library version. The statement #include <string.h>
  provides a default setting by which built-in functions are
  accessed. If you do not want the built-in function, you can use
  an #undef strcpy statement after including the file string.h.

Portability

   ANSI

Returns

  This function returns a pointer that is the same as the
  destination pointer.

See Also

   stccpy , stpcpy , strncpy

1.298 strcspn()

```
strcspn-Count the number of string characters not in the set

Synopsis

  #include <string.h>

  length = strcspn(s,b);

  size_t length;      /* span length in bytes */
  const char *s;      /* points to string being scanned */
  const char *b;      /* points to character set string */

Description

  This function measures the number of characters at the beginning
  of input string s that are not in the character set specified by
  the argument b.

Portability

   ANSI

Returns

  This function returns the number of bytes that are not in the
  specified character set.  The scan always stops when the NULL
  terminator byte is reached.

Example

  #include <stdio.h>
  #include <string.h>

  char *test = "This is a test";

  void checkspan(char *string, char *set)
  {
      printf("String:   %s\nScan Set: %s\nstrcspn:  %d\n",
       string, set, strcspn(string, set));
  }

  void main(void)
  {
      checkspan(test,"xyz");
      checkspan(test,"s");
      checkspan(test,"TXI");
  }

See Also

   stcis ,  stcisn ,  strspn
```

1.299 strdup()

strdup-Duplicate a string

Synopsis

```
  #include <string.h>

  p = strdup(s);

  char *p;     /* points to duplicate string */
  const char *s;     /* points to string being duplicated */
```

Description

This function creates a duplicate of the specified string by using the malloc and strcpy functions to allocate space and copy the string to it.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

XENIX

Returns

A NULL pointer is returned if the malloc functions fails. Otherwise, the function returns a pointer to the duplicate string.

See Also

malloc , strcpy

1.300 strerror()

strerror-Print the text for a given error number

Synopsis

```
  #include <string.h>

  p = strerror(error);

  char *p;       /* Pointer to text string   */
  int error;       /* Error number     */
```

Description

This function takes a specified error number and returns a pointer to a text string that describes the error.

Portability

ANSI

```
Returns

  This function returns a pointer to the text of the corresponding
  error message.  If it could not find the error, it returns a NULL
  pointer.  The string is valid until the next call to the strerror
  function and must not be modified by the caller.

Example

  /*
   * Print out an error from unlinking a file
   */
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>

  extern long errno;

  void main(void)
  {
      unlink("xyzzy:lambda");
      if (errno)
      {
    printf("Error removing file: %s\n", strerror(errno));
    exit(EXIT_FAILURE);
      }
  }

See Also

   errno
```

1.301 strftime()

```
strftime-Format a time string

Synopsis

  #include <time.h>

  ret = strftime(s, maxsize, format, timeptr);

  size_t ret;      /* number of characters in      */
          /*   formatted string       */
  char *s;      /* string to place characters in  */
  size_t maxsize;     /* maximum string size      */
  const char *format;   /* format instructions for string */
  const struct tm *timeptr; /* broken-down time information   */

Description

  This function is similar to the  sprintf  function but has its own
  formatting instructions for printing out time information.
  This function places characters into the array pointed to by the
  argument s in the format specified by the string pointed to by the
  argument format.  The format argument consists of zero or more
  conversion specifiers and ordinary characters.  All ordinary
  characters, including the terminating NULL character, are copied
  unchanged into the array, but the conversion specifiers are
  replaced by the appropriate characters. A conversion specifier
  consists of a percent (%) character followed by a character. The
  following list describes the characters with which each conversion
  specifier is replaced.

  Conversion
  Specifier   Replaced with . . .
  ----------  -------------------
     %a     the locale's abbreviated weekday name
     %A     the locale's full weekday name
     %b     the locale's abbreviated month name
     %B     the locale's full month name
     %c     the locale's appropriate date and time representation
     %d     the day of the month as a decimal number (01-31)
     %H     the hour (24-hour clock) as a decimal number (00-23)
     %I     the hour (12-hour clock) as a decimal number (00-12)
     %j     the day of the year as a decimal number (001-366)
     %m     the month as a decimal number (01-12)
     %M     the minute as a decimal number (00-59)
     %p     the locale's equivalent of the AM and PM designations
        associated with a 12-hour clock
     %S     the second as a decimal number (00-61)
     %U     the week number of the year (the first Sunday as the
        first day of week 1) as a decimal number (00-53)
     %w     the weekday as a decimal number (0-6), where Sunday is 0
     %W     the week number of the year (the first Monday as the
        first day of week 1) as a decimal number (00-53)
     %x     the locale's appropriate date representation
     %X     the locale's appropriate time representation
     %y     the year without the century as a decimal number (00-99)
     %Y     the year with the century as a decimal number
     %Z     the time zone name or abbreviation; no characters
        indicates the time zone is not determinable
     %%     a percent sign.

  No more than maxsize characters are placed into the array.  The
  appropriate characters are determined by the LC_TIME category of
  the current locale and by the values contained in the structure
  pointed to by the argument timeptr.  If copying takes place
  between objects that overlap or if the conversion specifier is not
  one of those listed above, the behavior is undefined.

Portability

   ANSI

Returns

  This function returns the number of characters placed into the
  string pointed to by the argument s, not including the terminating
  NULL character.  Otherwise, the strftime function returns a 0, and
```

the contents of the s argument are indeterminate.

See Also

asctime , gmtime , localtime , setlocale

1.302 stricmp()

stricmp-Compare strings, case insensitive

Synopsis

```
#include <string.h>

x = stricmp(a,b);

int x;          /* comparison result */
const char *a,*b;     /* strings being compared */
```

Description

This function compares two NULL-terminated strings using the ASCII collating sequence, but does not distinguish between uppercase and lowercase.

The relative collating sequence of the strings is indicated by the sign of the return value, as follows:

```
    Sign      Meaning
    ----      -------
    negative  first string is below the second
    zero      strings are equal
    positive  first string is above the second
```

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Returns

The sign of the return value indicates the relative collating sequence of the strings, as indicated above.

Example

```
#include <stdio.h>
#include <string.h>

void result(char *name, int r)
{
    char *p;
```

```
      if (r == 0)
      {
    p = "is equal to";
      }
      if (r < 0)
      {
    p = "is less than";
      }
      if (r > 0)
      {
    p = "is greater than";
      }
      printf("%s string A %s string B\n",name,p);
  }

  void main(void)
  {
      char a[256], b[256];

      while(1)
      {
    printf("Enter string A: ");
    if (fgets(a,sizeof(a),stdin) == NULL)
    {
        break;
    }
    printf("Enter string B: ");
    if (fgets(b,sizeof(b),stdin) == NULL)
    {
        break;
    }
    result("stricmp: ",stricmp(a,b));
      }
      printf("\n");
  }
```

See Also

strcmp , strncmp , strnicmp

1.303 strins()

strins-Insert a string

Synopsis

```
  #include <string.h>

  void strins(to,from);

  char *to;       /* destination string */
  const char *from;    /* source string    */
```

Description

```
  This function inserts the source string in front of the
  destination.  Both strings must be NULL-terminated, and the
  destination is shifted to the right (upward in memory) to
  accommodate the source string.  The final result is a single
  NULL-terminated string.

  Make sure that the destination area is large enough to hold both
  strings.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

```
   SAS/C
```

Example

```
  #include <stdio.h>
  #include <string.h>

  void main(void)
  {
      char *here = "Here ";
      char now[30];

      strcpy(now, "and now");
      printf("%s, %s\n", here, now);
      strins(now, here);        /* now => "Here and now" */
      printf("%s\n",now);
  }
```

See Also

```
   strcat
```

1.304 strlen()

strlen-Measure the length of a string

Synopsis

```
  #include <string.h>

  length = strlen(s);

  size_t length;   /* number of bytes in s (before null) */
  const char *s;
```

Description

```
  This function returns the number of bytes in string s before the
  NULL terminator byte.

  The strlen function has a built-in version that is equivalent to
  the standard library version.  The statement #include <string.h>
  provides a default setting by which built-in functions are
  accessed.  If you do not want the built-in function, you can enter
  an #undef strlen statement after including the file string.h.
```

Portability

   ANSI

Returns

  This function returns the number of bytes in the string before the
  NULL byte.

Example

```
  x = strlen("abc");   /* x is 3 */
  x = strlen("");      /* x is 0 */
```

See Also

   stclen

1.305 strlwr()

strlwr-Convert a string to lowercase

Synopsis

```
  #include <string.h>

  p  = strlwr(s);

  char *p;    /* return pointer (same as s) */
  char *s;    /* string pointer */
```

Description

  This function converts all alphabetic characters in the specified
  NULL-terminated string to lowercase, according to the 7-bit ASCII
  collating sequence.

  This function is not available if the _STRICT_ANSI flag has been
  defined.

Portability

   XENIX

Returns

  This function returns the original string pointer.

See Also

stricmp , strupr , tolower , toupper

1.306 strmfe()

strmfe-Make a filename with an extension

Synopsis

```
#include <string.h>

void strmfe(newname,oldname,ext);

char *newname;    /* new file name */
const char *oldname;  /* old file name */
const char *ext;  /* extension */
```

Description

This function copies the old filename to the new name, deleting any extension. Then it appends the specified extension to the new filename, with an intervening period. For example:

```
    Old name    Extension  New name
    --------    ---------  --------
    df1:myprog.c     cc      df1:myprog.cc
    abc        o       abc.o
```

The newname area must be large enough to accept the filename string and the separator. A safe size is FMSIZE which is defined in the dos.h header file.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

See Also

strmfn , strmfp

1.307 strmfn()

strmfn-Make a filename from components

Synopsis

```
#include <string.h>

void strmfn(file,drive,path,node,ext)
```

```
  char *file;         /* file name pointer */
  const char *drive;      /* drive code pointer */
  const char *path;       /* directory path pointer */
  const char *node;       /* node pointer */
  const char *ext;        /* extension pointer */
```

Description

This function makes a filename from four possible components. The name is constructed as follows:

```
   drive:path/node.ext
```

If the drive pointer is not NULL, the drive pointer is moved to the area pointed to by the file argument. Then, a colon is inserted unless one is already there. Next, if the path pointer is not NULL, it is appended to file, and the directory separator specified by _SLASH is added if necessary. The node string is appended next, unless it is NULL. Finally, if the ext pointer is not NULL, a period is appended to file, followed by the ext string.

Make sure that the file pointer refers to an area which is large enough to hold the result.

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Example

```
  #include <stdio.h>
  #include <string.h>
  #include <dos.h>

  char buffer[FMSIZE];

  void main(void)
  {
      /* The next statements both place "abc/def/ghi" */
      /* into the buffer. */

      printf("'', 'abc/def', 'ghi', ''\n");
      strmfn(buffer,NULL,"abc/def","ghi",NULL);
      printf("result = %s\n\n", buffer);

      printf("'', 'abc/def/', 'ghi', ''\n");
      strmfn(buffer,NULL,"abc/def/","ghi",NULL);
      printf("result = %s\n\n", buffer);

      /* The next statements both generate "df0:myfile.str" */

      printf("'df0', '', 'myfile', 'str'\n");
      strmfn(buffer,"df0",NULL,"myfile","str");
      printf("result = %s\n\n", buffer);

      printf("'df0:', '', 'myfile', 'str'\n");
      strmfn(buffer,"df0:",NULL,"myfile","str");
      printf("result = %s\n\n", buffer);
   }
```

See Also

strmfe , strmfp

1.308 strmfp()

strmfp-Make a filename from the path or node

Synopsis

```
  #include <string.h>

  void strmfp(name,path,node);

  char *name;      /* file name */
  const char *path;   /* directory path */
  const char *node;   /* node */
```

Description

This function copies the path string (if it is not NULL) to the file name area and appends the _SLASH separator if the path string does not end with a slash or colon. Then, the node string is appended to the file name. _SLASH is an external character variable that defaults to a slash (/).

The name area must be large enough to accept the filename string.

This function is not available if the _STRICT_ANSI

See Also

__matherr

1.309 tell()

tell-Get the level 1 file position

Synopsis

```
  #include <fcntl.h>

  apos = tell(fh);

  long apos;  /* absolute file position */
  int fh;     /* file handle */
```

Description

The tell function is equivalent to:

```
   apos = lseek(fh,0L,1);
```

The tell function returns a file position that can be used in a subsequent call to the lseek function to restore the file to the position at the time of the tell call.

Portability

UNIX

Returns

This function returns -1L if an error occurs, in which case the external integers errno and _OSERR contain additional error information.

See Also

errno , ftell , lseek , open, _OSERR

1.310 telldir()

telldir-Get the directory position

Synopsis

```
  #include <sys/dir.h>

  loc = telldir(dfd);

  long loc;  /* current read position */
  DIR *dfd;  /* directory file descriptor */
```

Description

This routine returns the current read position for the given directory file descriptor. This position is where the readdir function would obtain the next directory entry if it were called. The loc argument is also the value that you would pass to the seekdir function if you wanted to return directly to this same position.

loc values are good only for the life of the directory file descriptor. If you close a directory and reopen it, the loc values are no longer valid.

Portability

   UNIX

Returns

  This function returns the location of the next directory entry.

See Also

   closedir , opendir , readdir , seekdir

1.311 time()

time-Get the system time in seconds

Synopsis

```
  #include <time.h>

  timeval = time(timeptr);

  time_t timeval;   /* time value */
  time_t *timeptr;  /* pointer to time value storage */
```

Description

  This function returns the current time expressed as the number of
  seconds since 00:00:00 Greenwich Mean Time, January 1, 1970. If
  the timeptr pointer is not NULL, the time value is also stored in
  that location.

Portability

   ANSI

Example

```
  #include <stdio.h>
  #include <time.h>

  void main(void)
  {
      long t;

      time(&t);
      printf("Current time is %s\n",ctime(&t));
  }
```

See Also

   asctime , ctime , gmtime , localtime , tzset

1.312 __timecvt()

```
__timecvt-Convert a time_t value to an AmigaDOS DateStamp

Synopsis

  #include <time.h>

  tp = __timecvt(t);

  struct DateStamp *tp; /* pointer to converted DateStamp */
  time_t  t;    /* time value to convert */

Description

  This function converts a date and time in the format that the  time
  function returns to the AmigaDOS DateStamp format.

Portability

   AmigaDOS

Returns

  This function returns a pointer to a DateStamp array.  This array
  is considered read only and is valid only until the next call to
  the __timecvt function.  __timecvt only deals with local time, so
  no timezone conversion is performed.

Example

  /*
   * Convert the current time.
   */
  #include <stdio.h>
  #include <time.h>
  #include <dos.h>

  void main(void)
  {
      struct DateStamp *event;

      event = __timecvt(time(NULL));

      printf("Days since 1JAN78      = %ld\n"
       "Minutes after midnight = %ld\n"
       "Ticks past the minute  = %ld\n",
       event->ds_Days, event->ds_Minute, event->ds_Tick);
  }

See Also

   mktime
```

1.313 timer()

```
timer-Get the system clock with microseconds

Synopsis

  #include <time.h>

  x = timer(clock);

  int x;
  unsigned int clock[2];

Description

  The timer function obtains the current setting of the system clock
  in the form of a two-integer array as follows:

     clock[0] => seconds
     clock[1] => microseconds

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   AmigaDOS

Returns

  If successful, this function returns a 0.  Otherwise, it returns
  -1.
```

1.314 __tinymain()

```
__tinymain-Special version of the __main function

Synopsis

  #include <stdlib.h>

  void __stdargs __tinymain(line);

  char *line;  /* ptr to command line that caused execution */

Description

  The __tinymain function is provided for compatibility with
  previous releases.  Use of __tinymain increases the size of your
  executable module.  Do not use __tinymain.  The functionality
  provided by __tinymain is now automatically provided by the
  regular startup.

Portability

   OLD
```

```
See Also

   main
```

1.315 tmpfile()

```
tmpfile-Open a temporary file stream

Synopsis

  #include <stdio.h>

  fp = tmpfile(void);

  FILE *fp;      /* pointer to temporary file stream */

Description

  This function opens a temporary file. The name of the temporary
  file is constructed from a combination of the first three
  characters of the program name, a string of characters based on
  the base stack pointer, and a sequential number (nn):

      T:progname.stack.T.nn

  The tmpfile function attempts to open files of increasing
  sequential numbers until it succeeds, or it has tried 99 times.
  If you do not have the argument T: assigned, the tmpfile function
  will never be able to create a temporary file.

Portability

   ANSI

Returns

  This function returns a file handle for a file that can be read or
  written.  When this file is closed, it is automatically deleted.

Example

  /*
   * Create a temporary file to hold a sort data set
   */
  #include <stdio.h>

  void sortfile(FILE *fpo, FILE *fpi)
  {
      /* Insert code for "World's Greatest File Sorter" */
      /* in this routine...           */
      return;
  }

  void main(void)
  {
      FILE *fp, *infp, *outfp;

      fp = tmpfile();
      if (fp == NULL)
      {
    printf("Can't create sort temp file\n");
    return;
      }
      sortfile(fp, infp);
      sortfile(fp, outfp);
      fclose(fp);
  }
```

See Also

   open , close , tmpnam

1.316 tmpnam()

tmpnam-Create a temporary filename

Synopsis

```
  #include <stdio.h>

  name = tmpnam(b);

  char *name;  /* pointer to temporary file name */
  char *b;     /* pointer to name buffer or NULL */
```

Description

This function creates a unique name that can be used for a temporary file. If the b pointer is not NULL, it should point to a buffer at least L_tmpnam bytes long, and the function returns that pointer. If b is NULL, then the function creates a buffer and returns a pointer to it. Subsequent calls to the tmpnam function may modify the buffer.

Portability

   ANSI

Returns

This function returns a pointer to a temporary filename.

See Also

   tmpfile

1.317 toascii()

```
toascii-Convert a character to ASCII

Synopsis

  #include <ctype.h>

  cc = toascii(c);

  int cc;  /* converted character */
  int c;   /* character to convert */

Description

  The toascii function resets all high-order bits, leaving only the
  lower seven.

  You can use either characters or integers as arguments, but the
  macro is defined only over the integer range from -1 to 255.  The
  function, however, will return a result for values above 255, but
  the results are not necessarily correct and cannot be relied upon.
  The reason -1 is included as a valid argument is to avoid a
  nonsensical result if you feed the EOF value to one of the macros
  or functions.  EOF can be returned by the  getchar  function and
  other I/O functions, and if you pass it to any of the character
  test functions, the return value will be 0.

  If you include the file ctype.h as shown above, this function is
  actually defined as a macro and produces inline code to perform
  the conversion.  Without the ctype.h file, it is an actual
  function resolved in the standard library.  If you want to use the
  function version but must include the file ctype.h for some other
  reason, use an #undef statement to undefine the macros after
  including the ctype.h file.

      #undef toascii

  The toascii function is not available if the  _STRICT_ANSI  flag has
  been defined.

Portability

   SAS/C

See Also

   __ctype ,  isascii
```

1.318 tolower()

```
tolower-Convert a character to lowercase

Synopsis
```

```
  #include <ctype.h>

  cc = tolower(c);

  int cc;  /* converted character */
  int c;   /* character to convert */
```

Description

The tolower function tests if the argument c is an uppercase alphabetic character and, if so, converts it to lowercase.

You can use either characters or integers as arguments, but the macro is defined only over the integer range from -1 to 255. The function, however, will return a result for values above 255, but the results are not necessarily correct and cannot be relied upon. The reason -1 is included as a valid argument is to avoid a nonsensical result if you feed the EOF value to one of the macros or functions. EOF can be returned by the getchar function and other I/O functions, and if you pass it to any of the character test functions, the return value will be 0.

If you include the file ctype.h as shown above, this function is actually defined as a macro and produces inline code to perform the conversion. Without the ctype.h file, it is an actual function resolved in the standard library. If you want to use the function version but must include the file ctype.h for some other reason, use an #undef statement to undefine the macros after including the ctype.h file.

```
      #undef tolower
```

Portability

ANSI

See Also

__ctype , islower

1.319 toupper()

toupper-Convert a character to uppercase

Synopsis

```
  #include <ctype.h>

  cc = toupper(c);

  int cc;  /* converted character */
  int c;   /* character to convert */
```

Description

The toupper function is the reverse of the tolower function.

You can use either characters or integers as arguments, but the macro is defined only over the integer range from -1 to 255. The function, however, will return a result for values above 255, but the results are not necessarily correct and cannot be relied upon. The reason -1 is included as a valid argument is to avoid a nonsensical result if you feed the EOF value to one of the macros or functions. EOF can be returned by the getchar function and other I/O functions, and if you pass it to any of the character test functions, the return value will be 0.

If you include the file ctype.h as shown above, this function is actually defined as a macro and produces inline code to perform the conversion. Without the ctype.h file, it is an actual function resolved in the standard library. If you want to use the function version but must include the file ctype.h for some other reason, use an #undef statement to undefine the macros after including the ctype.h file.

```
    #undef toupper
```

Portability

ANSI

Example

```
/*
 * The following program echoes each input
 * line in upper case.
 */
#include <stdio.h>
#include <ctype.h>

void main(void)
{
    char b[256],*p;

    while(gets(b) != NULL)
    {
  for (p = b; *p != '\0'; p++)
  {
      *p = toupper(*p);
  }
  puts(b);
    }
}
```

See Also

__ctype , isupper

1.320 tqsort()

```
tqsort-Sort an array of text pointers

Synopsis

  #include <stdlib.h>

  void tqsort(ta,n);

  char *ta[];   /* pointer to text pointer array */
  size_t n;   /* number of elements in array */

Description

  The tqsort function sorts the specified data array using the ACM
  271 algorithm, more popularly known as Quicksort.  During its
  operation, it calls on the  strcmp  comparison routine with pointers
  to the two array elements being compared.

  The ta array consists of pointers to NULL-terminated character
  strings.  This function rearranges the pointers so that the
  strings are in ascending ASCII sequence. The sort is based on the
  contents of the strings rather than their physical address.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   SAS/C

See Also

   dqsort ,  fqsort ,  lqsort ,  qsort ,  sqsort
```

1.321 __tzset()

```
__tzset-Set the time zone variables

Synopsis

  #include <time.h>

  void __tzset(void);

     /* If the _STRICT_ANSI flag has not been defined,
     * these symbols are defined in time.h:
     * extern int __daylight;
     * extern long __timezone;
     * extern char *__tzname[2];
     * extern char __tzstn[4];
     * extern char __tzdtn[4];
     * extern char *_TZ;
     */
```

Description

```
  The __tzset function assigns values to the time zone variables
   __daylight ,  __timezone , and  __tzname . These variables are then
  used by the  localtime  function and other functions to correct from
  Greenwich Mean Time (GMT) to local time.

  The values for these variables are obtained from the character
  string pointer named  _TZ , which has the following form:

      char *_TZ = "aaabbbccc"

  aaa is the three-letter abbreviation for the local standard time
  zone (for example, CST), and bbb is a number from -23 to 24 that
  specifies the number to be subtracted from GMT to obtain local
  standard time.  Both aaa and bbb are required.  ccc is the
  abbreviation for the local daylight savings time zone (for
  example, CDT), and it should be present only if daylight savings
  time is currently in effect.

  Initially, the  _TZ  pointer is set to NULL.  It should be
  initialized with the address of a string corresponding to the
  correct time zone.  If  _TZ  is NULL, the __tzset function uses the
  default string CDT6.

  When the __tzset function is called, the  __timezone  integer is
  loaded with the number of seconds that must be subtracted from GMT
  to get the local time.  Next, the  __daylight  integer is loaded
  with 0 if the ccc portion of the  _TZ  pointer is absent and 1 if
  ccc is present.  Then, the aaa and ccc parts are copied to  __tzstn
  and  __tzdtn , respectively, with NULL terminators.  Finally,
   __tzname [0] and  __tzname [1] are loaded with pointers to  __tzstn
  and  __tzdtn , respectively.

  The symbols defined in the file time.h are not available if the
   _STRICT_ANSI  flag has been defined.
```

Portability

```
   XENIX
```

See Also

~ localtime

1.322 ungetc()

ungetc-Push input character back

Synopsis

```
  #include <stdio.h>
  r = ungetc(c, fp);

  int r;      /* return character or code */
  int c;      /* character to be pushed back */
  FILE *fp;   /* file pointer */
```

Description

This function pushes a character back to the specified level 2 input file. The character need not be the same as the one that was most recently read. However, before calling the ungetc function, you must have read at least one character using the fgetc function or one of the other level 2 input functions. Also, you can only push back one character; if you call the ungetc function more than once between input functions, the results are undefined.

Portability

ANSI

Returns

Normally, the ungetc function returns the character that was pushed back. However, if the end-of-file has been reached or if no characters have been read yet, the value EOF is returned.

Example

```
  #include <stdio.h>

  void main(void)
  {
      int c;

      while(1)
      {
    printf("Loop 1...\n");
    while((c = getchar()) != EOF)
    {
        if (isalpha(c))
        {
      putchar(c);
        }
        else
        {
      break;
        }
    }
    ungetc(c, stdin);
    printf("\n\nLoop 2...\n");
    while((c = getchar()) != EOF)
    {
        if (isalpha(c) == 0)
        {
      putchar(c);
        }
        else
        {
      break;
        }
    }
    ungetc(c, stdin);
      }
      printf("\n\nDone\n");
  }
```

1.323 unlink()

```
unlink-Remove a file

Synopsis

  #include <stdio.h>

  error = unlink(name);

  int error;      /* non-zero if error */
  const char *name;   /* file name */

Description

  This function removes the specified file from the system. The file
  name argument can include a path, but it cannot include wild card
  characters.  That is, you can remove only one file at a time.

  The unlink function is provided for compatibility with some
  versions of UNIX.

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.

Portability

   UNIX

Returns

  If a nonzero value is returned, some type of error occurred, and
  additional information can be found in the external integers  errno
  and  _OSERR .  The most common errors occur when you try to remove a
  file that doesn't exist, that is marked as read-only, or that is
  in use.

Example

  /*
   * This program removes all files specified
   * in the argument list.  It does not allow
   * wild card characters in the file names.
   */
  #include <stdio.h>
  #include <stdlib.h>

  void main(int argc, char *argv[])
  {
      int i;      /* loop counter */
      /* exit code, non-zero if any failures */

      int ret = EXIT_SUCCESS;

      for (i = 1; i < argc; i++)
      {
    if (unlink(argv[i]))
    {
        perror("RMV");
        ret = EXIT_FAILURE;
    }
      }
      exit(ret);
  }
```

```
See Also

   errno ,  _OSERR ,  remove
```

1.324 utpack()

```
utpack-Pack UNIX time

Synopsis

  #include <time.h>

  ut = utpack(x);

  long ut;   /* packed UNIX time */
  const char *x;   /* unpacked UNIX time */

Description

  This function packs the 32-bit time value that is traditionally
  used in UNIX systems.  This value is the number of seconds since
  00:00:00, January 1, 1970.  The  time  function returns the system
  clock in this form relative to Greenwich Mean Time.

  The unpacked time is a 6-byte array in the following format:

      Byte  Contents
      ----  --------
      x[0]  year - 1970 (-128 to 127)
      x[1]  month (1 to 12)
      x[2]  day (1 to 31)
      x[3]  hour (0 to 23)
      x[4]  minute (0 to 59)
      x[5]  second (0 to 59)
  Although this array is similar to the one produced by the  getclk
  function and used by the  stpdate  function, the year is biased
  relative to 1970 instead of 1980.  The year is a signed character
  and can be negative.  A value of -3, for example, is 1967 (in
  other words, 1970 - 3).

  This function is not available if the  _STRICT_ANSI  flag has been
  defined.
```

Portability

SAS/C

Returns

This function returns a packed long integer, as described above.

Example

```
/*
 * Get a file time and subtract 10 years from it.
 * No error checks.
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <dos.h>

void main(int argc, char *argv[])
{
    char tt[6];
    long ft,ut;

    ft = getft(argv[1]);
    utunpk(ft, tt);
    tt[0] -= 10;
    ut = utpack(tt);
    printf("File time is: %s\n",ctime(&ut));
}
```

See Also

ctime , getclk , gmtime , localtime , stpdate , time , utunpk

1.325 utunpk()

utunpk-Unpack UNIX time

Synopsis

```
#include <time.h>

void utunpk(ut,x);

long ut;  /* packed UNIX time */
```

```
  char *x;  /* unpacked UNIX time */
```

Description

This function unpacks the 32-bit time value that is traditionally used in UNIX systems. This value is the number of seconds since 00:00:00, January 1, 1970. The time function returns the system clock in this form relative to Greenwich Mean Time.

The unpacked time is a 6-byte array in the following format:

```
    Byte  Contents
    ----  --------
    x[0]  year - 1970 (-128 to 127)
    x[1]  month (1 to 12)
    x[2]  day (1 to 31)
    x[3]  hour (0 to 23)
    x[4]  minute (0 to 59)
    x[5]  second (0 to 59)
```

Although this array is similar to the one produced by the getclk function and used by the stpdate function, the year is biased relative to 1970 instead of 1980. So, if you use the utunpk function followed by the stpdate function, you must subtract 10 from x[0] before the stpdate call. The year is a signed character and can be negative. A value of -3, for example, is 1967 (in other words, 1970 - 3).

This function is not available if the _STRICT_ANSI flag has been defined.

Portability

SAS/C

Example

```
/*
 * Get a file time and subtract 10 years from it.
 * No error checks.
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <dos.h>

void main(int argc, char *argv[])
{
    char tt[6];
    long ft,ut;

    ft = getft(argv[1]);
    utunpk(ft, tt);
    tt[0] -= 10;
    ut = utpack(tt);
    printf("File time is: %s\n",ctime(&ut));
}
```

See Also

ctime , getclk , gmtime , localtime , stpdate , time , utpack

1.326 va_arg()

va_arg-Get an argument from a varying-length argument list

Synopsis

```
#include <stdarg.h>

(arg_type) va_arg(va_list ap, arg_type);
```

Description

The va_arg macro returns the value of the next argument in a varying-length argument list.

The first argument, ap, is a work area of type va_list, which is used by various macros defined in the file stdarg.h. (The va_list must be initialized by a previous use of the va_start macro, and a corresponding va_end should be called when processing of the arguments is finished.

The second argument, arg_type, is the type of the argument that is expected. The arg_type argument must be written in such a form that arg_type * is the type of a pointer to an element of that type. For example, char is a valid arg_type because char * is the type of a pointer to a character. int(*)() is not a valid second argument to the va_arg macro because int(*)()* is not a valid type. (You can use typedef declarations to create usable synonyms of this sort for any type.)

The results of the va_arg macro are unpredictable if the argument values are not appropriate.

In certain cases, arguments are converted when they are passed to another type. For instance, char and short arguments are converted to int, float to double, and array to pointer. When parameters of this sort are expected, the va_arg macro must be issued with the type after conversion. For example, va_arg(ap, float) may fail to access a float argument value correctly, so va_arg(ap, double) should be used.

NOTE: There is no way to test whether a particular argument is the last one in the list. Attempting to access arguments after the last one in the list produces unpredictable results.

Portability

ANSI

Returns

```
  The va_arg macro returns the value of the next argument in the
  list.  The type is always the same as the second argument to the
  va_arg macro.
```

Example

```
  /*
   * This example shows a function named concat,
   * which can be used to concatenate any number
   * of strings.  A simple call is concat(3,a,b,c).
   * This should have the same effect as
   *
   *   strcat (a,b);
   *   strcat (a,c);
   *
   * The first argument is the total number of strings.
   */

  #include <stdarg.h>
  #include <string.h>
  #include <stdio.h>

  void concat (int count, ...);

  void main(void)
  {
      char str[20] = "abcd";

      concat(4,str,"efgh","ijkl","mnop");

      printf("The concatenated string = %s\n",str);
  }

  void concat (int count, ...)
  {
      va_list ap;
      char *target, *source;

      if (count <=1)
    return;

      va_start(ap, count);

      /* Get target string */
      target = va_arg (ap, char *);

      /* Point to string end */
      target += strlen(target);

      while (--count > 0)
      {
    /* Get next source string */
    source = va_arg(ap, char *);

    /* Copy chars to target */
    while (*source)
```

```
        *target++ = *source++;
      }

      /* Add final null */
      *target = '\0';

      /* End arg list processing */
      return;
  }
```

See Also

   va_end , va_start

1.327 va_end()

va_end-End varying-length argument list processing

Synopsis

```
  #include <stdarg.h>

  void va_end(va_list ap);
```

Description

  The va_end macro completes processing of a varying-length argument
  list. The argument ap is a work area of type va_list, which is
  used by various macros defined in the file stdarg.h.

  After the va_end macro is called, the va_start macro must be
  called again before the va_arg macro can be used.

  In this implementation, use of the va_end macro in varying-length
  argument list processing is not required. However, in other
  implementations, failure to issue the va_end macro may cause
  program failures on return from the function that issued the
   va_start macro.

Portability

   ANSI

Example

  See the example for the va_arg macro.

See Also

   va_arg , va_start

1.328 va_start()

```
va_start-Begin varying-length argument list processing

Synopsis

  #include <stdarg.h>

  void va_start(va_list ap, arg_name);

Description

  The va_start macro initializes processing of a varying-length
  argument list.  The first argument, ap, is a work area of type
  va_list, which is used by various macros defined in the file
  stdarg.h. The second argument, arg_name, is the name of the
  parameter to the calling function after which the varying part of
  the parameter list begins (the parameter immediately before the
  ,...).

  The results of the va_start macro are unpredictable if the
  argument values are not appropriate.

Portability

   ANSI

Example

  See the example for the  va_arg  macro.

See Also

   va_arg ,  va_end
```

1.329 vfprintf()

```
vfprintf-Formatted write of a varying-length argument list to a file

Synopsis

  #include <stdarg.h>
  #include <stdio.h>

  n = vfprintf(fp, ctl, args);

  int n;      /* number of characters written   */
        /*   or -1 for error        */
  FILE *fp;   /* file to be written to      */
  const char *ctl;  /* control string specifying formatting */
  va_list args;   /* items to be formatted      */

Description

  This function is identical in capabilities to the  fprintf
  function, except that the argument list is passed as a va_list
  instead of on the stack.  The argument list args must be
  initialized by the caller with a  va_start  macro (and any preceding
   va_arg  macros that it wants to call).  When terminated, it is the
  responsibility of the caller to call the  va_end  macro on the
  argument list.
```

Portability

   ANSI

Returns

  This function returns the number of characters written or, in the
  case of an error, a -1.

Example

```
  #include <stdio.h>
  #include <stdarg.h>

  /* My own error function for a given error number */
  void myerr(FILE *fp, int ernum, char *string, ...)
  {
      va_list arglist;

      va_start(arglist, string);

      fprintf(fp,"ERR-%d: \n",ernum);
      vfprintf(fp, string, arglist);

      va_end(arglist);
  }

  void main(void)
  {
      myerr(stderr, 205, __sys_errlist[205]);
  }
```

See Also

   fprintf

1.330 vprintf()

vprintf-Formatted write of a varying-length argument list to standard
output

Synopsis

```
  #include <stdarg.h>
  #include <stdio.h>

  x = vprintf(ctl, args);

  int x;      /* number of characters written   */
```

```
       /*   or -1 for error        */
  const char *ctl;  /* control string specifying formatting */
  va_list args;   /* items to be formatted      */
```

Description

  This function is identical in capabilities to the printf function,
  except that the argument list is passed as a va_list instead of on
  the stack. The argument list args must be initialized by the
  caller with a va_start macro (and any preceding va_arg macros
  that it wants to call). When terminated, it is the
  responsibility of the caller to call the va_end macro on the
  argument list.

Portability

   ANSI

Returns

  This function returns the number of characters written or, in the
  case of an error, a -1.

Example

```
  #include <stdio.h>
  #include <stdarg.h>

  /* My own error function for a given error number */
  void myerr(int ernum, char *string, ...)
  {
      va_list arglist;

      va_start(arglist, string);

      printf("ERR-%d: ",ernum);
      vprintf(string, arglist);

      va_end(arglist);
  }

  void main(void)
  {
      myerr(205, _sys_errlist[205]);
  }
```

See Also

   printf

1.331 vsprintf()

vsprintf-Formatted write of a varying-length argument list to a string

Synopsis

```
  #include <stdarg.h>
  #include <stdio.h>

  x = vsprintf(buf, ctl, args);

  int x;      /* number of characters placed in the   */
        /*   output buffer        */
  char *buf;    /* String for resulting image     */
  const char *ctl;  /* control string specifying formatting */
  va_list args;   /* items to be formatted      */
```

Description

This function is identical in capabilities to the sprintf function, except that the argument list is passed as a va_list instead of on the stack. The argument list args must be initialized by the caller with a va_start macro (and any preceding va_arg macros that it wants to call). When terminated, it is the responsibility of the caller to call the va_end macro on the argument list.

Portability

ANSI

Returns

This function returns the number of characters placed in the output buffer (excluding the terminating NULL byte).

Example

```
  #include <stdio.h>
  #include <stdarg.h>

  /* Format an arbitrary message into a buffer */
  void getmsg(char *buffer, char *string, ...)
  {
      va_list arglist;

      va_start(arglist, string);

      vsprintf(buffer, string, arglist);

      va_end(arglist);
  }

  void main(void)
  {
      char buf[256];

      getmsg(buf, "Formatted with %d argument.\n", 1);
      printf(buf);
  }
```

See Also

sprintf

1.332 wait()

wait-Wait for single child process to complete

Synopsis

```
#include <dos.h>

cc = wait(procid);

int cc;       /* child's completion code         */
struct ProcID *procid;    /* pointer to process ID structure */
```

Description

After a process creates a child with the fork and forkv functions, the parent continues to execute until it calls either the wait or waitm function; that is, the parent and child are multiprogrammed. When the child process completes, the parent process can get its completion code with the wait or waitm function.

See the description of the forkl and forkv functions for more information.

Portability

SAS/C

Returns

This function returns the integer completion code of the child process.

Example

See the example under the forkl and forkv functions.

See Also

exit , forkl , forkv , AmigaDOS functions LoadSeg and CreateProc (The AmigaDOS Manual, 3rd Edition)

1.333 waitm()

waitm-Wait for multiple child processes to complete

Synopsis

```
#include <dos.h>
```

```
  complist = waitm(proclist);

  struct ProcID *complist;  /* pointer to linked list of       */
          /*  completed process IDs      */
  struct ProcID **proclist; /* address of pointer to linked    */
          /*  list of process IDs      */
```

Description

After a process creates a child with the fork and forkv functions, the parent continues to execute until it calls either the wait or waitm function; that is, the parent and child are multiprogrammed. When the child process completes, the parent process can get its completion code with the wait or waitm function.

See the description of the forkl and forkv functions for more information.

Portability

SAS/C

Returns

This function returns the a pointer to a linked list of completed child process IDs.

Example

See the example under the forkl and forkv functions.

See Also

exit , forkl , forkv , AmigaDOS functions LoadSeg and CreateProc (The AmigaDOS Manual, 3rd Edition)

1.334 wcstombs()

wcstombs-Convert a wide-character string to a multibyte string

Synopsis

```
  #include <stdlib.h>

  length = wcstombs(s, pwcs, n);

  size_t length;    /* length or state information */
  char *s;    /* pointer to characters */
  const wchar_t *pwcs;  /* pointer to wide-character string */
  size_t n;   /* maximum number of characters to look at */
```

Description

This function converts a NULL-terminated wide-character string to

```
  a NULL-terminated multibyte string. The wcstombs function for the
  current locale is passed the input parameters, and the result is
  returned.

Portability

   ANSI

See Also

   mblen ,  wctomb
```

1.335 wctomb()

```
wctomb-Map a wide character to a multibyte character

Synopsis

  #include <stdlib.h>

  length = wctomb(s, wc);

  int length;     /* length or state information */
  char *s;        /* pointer to characters or NULL */
  wchar_t wc;     /* wide character */

Description

  This function converts a wide character to a multibyte character
  sequence.  The wctomb function for the current locale is passed
  the input parameters, and the result is returned.

Portability

   ANSI

See Also

   mblen ,  wcstombs
```

1.336 write()

```
write-Write to a level 1 file

Synopsis

  #include <fcntl.h>

  count = write(fh,buffer,length);

  int count;         /* actual bytes read or written */
  int fh;          /* file handle */
  void *buffer;       /* data buffer */
  unsigned int length;  /* number of bytes to read or write */
```

Description

  This function writes a level 1 file whose handle was returned by
  the creat or open function. Under normal circumstances, the value
  returned should match the buffer length. If this value is -1 or
  greater than the requested length, then some type of error
  occurred, and you should consult the external integers errno and
   _OSERR .

  If the actual length is less than the requested length when
  reading, this usually means that the file is exhausted.
  Similarly, if the actual length is less than the requested length
  for a write operation, this usually means that the device has no
  more space available. In both of these cases, it is still a good
  idea to check the external integers errno and _OSERR just in case
  some malfunction caused the short count.

  Level 1 files are automatically closed by the exit function, which
  is usually called for you when the program terminates.

Portability

   UNIX

Returns

  If the operation is successful, this function returns the actual
  number of bytes transferred. Otherwise, it returns -1 and places
  error information in the external integers errno and _OSERR .

See Also

   errno , fwrite , open , _OSERR , read

1.337 _xcexit()

_XCEXIT-Terminate the program

Synopsis

```
  #include <stdlib.h>

  void _XCEXIT(lcode);

  long lcode;
```

Description

  This function terminates execution of the current program and
  returns control to the parent program.

  _XCEXIT calls the standard termination routines. The parameter

```
  lcode is a value from 0 to 255 that gets passed back to the
  parent.  By convention, a value of 0 indicates success.

  Normally, your program should call the  exit  or  __exit  functions.

  _XCEXIT is a symbol defined in the startup code, so it does not
  exist when you use a startup file or shared library that you have
  created.

  If you are linking a shared library and you get a reference to
  XCEXIT as an undefined symbol, then you are linking in code that
  is attempting to call either  exit ,  abort , or another function
  that makes the program terminate.  From within a shared library,
  you must not call any library functions that terminate your
  program.  For example, you cannot call  exit ,  __exit , or  abort
  from a shared library.  You also cannot use  setjmp  and  longjmp  to
  jump across a call from the program into the library.

Portability

   SAS/C

See Also

   exit ,  __exit
```

1.338 cppclass

```
 class complex
 abs(), arg(), conj(), imag(), norm(), polar(), real()
 exp(), log(), pow(), sqrt()
 sin(), cos(), sinh(), cosh()
 Complex Operators
 <<, >>
 class fstream
 class ifstream
 class ios
 class ios, enum format_state
 class ios, enum io_state
 class ios, enum open_mode
 class ios, enum seek_dir
 class iostream
 class istream
 class istrstream
 class ofstream
 class ostream
 class ostrstream
 class stdiostream
 class streampos
 class strstream
 class filebuf
 class stdiobuf
 class streambuf
 class strstreambuf
 class IOMANIP
```

1.339 class complex

```
class complex-Constructors and conversion operators

Synopsis

  #include <complex.h>

  class complex
  {
  public:
     complex();
     complex(double real, double imag = 0.0);
  };

Description

  The following constructors are defined for class complex:

      complex();
    enables you to declare complex variables without
    initializing them. static and external complex variables
    declared without an initializer have an initial value of
    (0,0); other uninitialized complex variables have an
    undefined initial value.

      complex(double real, double imag = 0.0);
    allows explicit initialization of complex variables. For
    example, the following two statements are valid:

        complex c1(1.0, 2.0);
        complex c2(1.0);   // The imaginary part is 0.0.

    This constructor also allows for implicit conversion from
    arithmetic types to complex values. For example, the
    following two statements are valid:

        complex c3 = 3.4; // c3 is (3.4, 0.0).
        c3=10;   // c3 is (10.0, 0.0).

    Using this constructor, you can also create complex values
    within expressions.  For example:

         c2 = c3 + complex(1.2, 3.5);   // Uses complex::operator +.

    The temporary object created by the expression
    complex(1.2,3.5) gets destroyed.
```

1.340 abs(), arg(), conj(), imag(), norm(), polar(), real()

```
abs(), arg(), conj(), imag(), norm(), polar(), real()-Cartesian and polar
coordinate functions

Synopsis

  #include <complex.h>

  class complex
  {
  public:
      friend double abs(complex c);
      friend double arg(complex c);
      friend complex conj(complex c);
      friend double imag(complex c);
      friend double norm(complex c);
      friend complex polar(double r, double t);
      friend double real(complex c);
  };

Description

  The following list describes the functions defined for class
  complex.  In these descriptions, d, r, and t are of type double,
  and c and z are of type complex.

      d = abs(c)
    returns the absolute value (magnitude) of c.

      d = arg(c)
    returns the angle of c (measured in radians) in the
    half-open interval (-pi to pi).

      z = conj(c)
    returns the conjugation of c.  If c is (x,y), then conj(c)
    is (x,-y).

      d = imag(c)
    returns the imaginary part of c.

      d = norm(c)
    returns the square of the magnitude of c.

      z = polar(r, t)
    returns a complex number.  The arguments represent a pair
    of polar coordinates where r is the magnitude and t is the
    angle (measured in radians).

      d = real(c)
    returns the real part of c.
```

1.341 exp(), log(), pow(), sqrt()

```
exp(), log(), pow(), sqrt()-Exponential, logarithmic, power, and square
root functions
```

Synopsis

```
  #include <complex.h>

  class complex
  {
  public:
      friend complex exp(complex c);
      friend complex log(complex c);
      friend complex pow(double c, complex b);
      friend complex pow(complex c, int b);
      friend complex pow(complex c, double b);
      friend complex pow(complex c, complex b);
      friend complex sqrt(complex c);
  };
```

Description

The following list describes the additional functions defined for class complex. These function names are overloaded by the C++ libraries. In these descriptions, z is of type complex, and c and b are of the types indicated by the function prototypes in the Synopsis.

    z = exp(c)
  returns e**c.

    z = log(c)
  returns the natural logarithm of c. When c is (0,0), log(c) returns (-HUGE,0), and errno is set to EDOM.

    z = pow(c, b)
  returns c**b.

    z = sqrt(c)
  returns the square root of c that is contained in the first or fourth quadrant of the complex plane.

Returns

In all overflow cases, errno is set to ERANGE.

For the log() function, if overflow is caused by the real part of c being small or the imaginary part of c being large, then exp(c) returns (0,0) and errno is set to ERANGE.

If the real part of c is large enough to cause overflow, exp(c) returns different values depending on the sine and cosine of the imaginary part of c. The real portion of a complex number c depends on the cos(imag(c)), and the imaginary part depends on the sin(imag(c)). The following table shows the values returned by exp(c) when the real part of c is large enough to cause overflow. HUGE corresponds to the largest representable double.

1.342 sin(), cos(), sinh(), cosh()

```
sin(), cos(), sinh(), cosh()-Trigonometric and hyperbolic functions

Synopsis

  #include <complex.h>

  class complex
  {
  public:
      friend complex sin(complex c);
      friend complex cos(complex c);
      friend complex sinh(complex c);
      friend complex cosh(complex c);
  };

Description

  The following list describes the trigonometric and hyperbolic
  functions defined for class complex.  In these descriptions, c and
  z are of type complex.

      z = sin(c)
    returns the sine of c.

      z = cos(c)
    returns the cosine of c.

      z = sinh(c)
    returns the hyperbolic sine of c.

      z = cosh(c)
    returns the hyperbolic cosine of c.

Returns

  In all overflow cases,  errno  is set to ERANGE.

  sin(c) and cos(c) return (0,0) if the real part of c causes
  overflow. If the imaginary part of c causes an overflow, sin(c)
  and cos(c) return values according to the tables in your Library
  Reference Manual.  HUGE corresponds to the largest representable
  double.

  sinh(c) and cosh(c) return (0,0) if the
  imaginary part of c causes overflow. If the real part of
  c causes an overflow, sinh(c) and cosh(c) return
  values according to the tables in your Library Reference Manual.
```

1.343 complex operators

```
Complex Operators-Operators for the C++ complex library
```

Synopsis

```
  #include <complex.h>

  class complex
  {
  public:
      friend complex operator +(complex c, complex b);
      friend complex operator -(complex c);
      friend complex operator -(complex c, complex b);
      friend complex operator *(complex c, complex b);
      friend complex operator /(complex c, complex b);
      friend complex operator /(complex c, double d);
      friend int operator ==(complex c, complex b);
      friend int operator !=(complex c, complex b);
      void operator +=(complex c);
      void operator -=(complex c);
      void operator *=(complex c);
      void operator /=(complex c);
      void operator /=(double d);
  };
```

Description

```
  The -, /, and /= operators are overloaded for complex numbers. The
  following list describes the function of each operator, in the
  order of precedence.  In these descriptions, c and b are of type
  complex, and d is of type double.

      -c
    is the arithmetic negation of c.

      c * b
    is the arithmetic product of c and b.

      c / b
    is the arithmetic quotient of c and b.

      c / d
    is the arithmetic quotient of c and d.

      c + b
    is the arithmetic sum of c and b.

      c - b
    is the arithmetic difference of c and b.

      c == b
    is nonzero if c is equal to b; otherwise, it is zero.

      c != b
    is nonzero if c is not equal to b; otherwise, it is zero.

      c += b
    assigns to c the arithmetic sum of itself and b.

      c -= b
    assigns to c the arithmetic difference of itself and b.

      c *= b
    assigns to c the arithmetic product of c and b.

      c /= b
    assign to c the arithmetic quotient of c and b.

      c /= d
    assign to y the arithmetic quotient of c and d.


  CAUTION:  The assignment operators do not yield a value that can
  be used in an expression.

  For example, the following construction is not valid:

      complex a, b, c;

      a = (b += c);
```

Example

```
  #include <complex.h>

  int main(void)
  {
      complex x;
      complex y(10.0,20.0);
      complex z(30.0,40.0);
      double d;

      x = y + z;

      cout << "x is " << x << endl;

      x = y*y;  // Get y squared

      cout << y << " squared is " << x << endl;

      if (x == y*y)
    cout << "Equality operator works" << endl;

      d = real(y);
      printf("real(y) == %g\n", d);

      return 0;
  }
```

1.344 <<, >>

<<, >>-Complex I/O insertion and extraction operators

Synopsis

```
  #include <complex.h>

  ostream& operator <<(ostream& os, complex c);
  istream& operator >>(istream& is, complex& c);
```

Description

  The following operators provide insertion and extraction
  capabilities for complex numbers.

```
      ostream& operator <<(ostream& os, complex c);
```
  writes a complex number c to the output stream os. The
  output is formatted as follows:

```
        (real-part,imaginary-part)
```

  Both parts are formatted as doubles. The formatting is
  controlled by flags associated with the stream. For more
  information, see the description of enum format_state and
  of ostream::operator <<(double) in the description of
  class ostream.

```
      istream& operator >>(istream& is, complex& c);
```
  reads a formatted complex number from is into c. The
  istream should contain the complex number to be read in
  one of these formats:

```
        (real-part,imaginary-part)
        (real-part)
```

  Both parts would be formatted as doubles. The formatting
  is controlled by flags associated with the stream. For
  more information, see the description of enum format_state
  and of istream::operator >>(double) in the description of
  class istream.

  Remember the following when performing complex I/O:

  - you must use the parentheses and comma for input

  - you can use white space in your input but it is not
    significant.

  If your input variable represents a real number such as 5e-2 or
  (502), the >> operator interprets it as a complex number with an
  imaginary part of 0.

Returns

  If the istream does not contain a properly formatted complex
  number, operator >> sets the ios::failbit bit in the stream's I/O
  state.

Examples

```
  // The following code writes the string
  // "This is a complex: (3.4,2.1)" to cout.

  complex c(3.4,2.1);
  cout << "This is a complex: " << c << endl;


  // If cin contains (1.2,3.4), then the following code
  // reads that value into c.

  complex c;
  cin >> c;
```

See Also

  class istream, class ostream, enum format_state

1.345 class fstream

class fstream-Provide formatted file I/O

Synopsis

```
  #include <fstream.h>

  class fstream : public iostream
  {
  public:
      fstream();
      fstream(const char *name, int mode);

      virtual ~fstream();

      void open(const char *name, int mode);
      void close();

      void setbuf(char *p, size_t len);
      filebuf* rdbuf();
  };
```

Description

  class fstream implements an input/output stream whose destination is a file. The streambuf associated with the I/O operations is a filebuf (instead of a strstreambuf or stdiobuf).

Parent Class

  class fstream inherits characteristics from class iostream, which inherits characteristics from class ios. See the descriptions of these parent classes for the details on functions and operators that are inherited.

Constructors

  There are two sets of constructors for class fstream, as follows:

```
      fstream::fstream();
    creates an unopened stream of the appropriate type.

      fstream::fstream(const char *name, int mode);
    creates a stream of the appropriate type, named name,
    using the specified mode.

    See the description of enum open_mode, later in this
    chapter, for information on the available modes. If the
    open fails, the stream's status is reflected in its I/O
    state flags.  See the description of enum io_state for
    information on the I/O state flags.

Destructors

  class fstream has one destructor:

      virtual fstream::~fstream();
    close the stream, if opened.

Member Functions

  The following descriptions give the purpose and return type of the
  member functions, as well as any other appropriate information.

      void fstream::open(const char *name, int mode);
    opens the named file using the specified mode.  There is
    no default mode bit set.  See the description of enum
    open_mode for information on the available open modes.

    This function does not have a return value. If an error
    occurs during the open, the ios::failbit bit is set in the
    stream's I/O state.  For example, the file may already
    exist, or the call to rdbuf()->open() may fail.

      void fstream::close();
    closes the connection between the appropriate object and
    its associated file.  Unless an error occurs, all bits in
    the object's I/O state are set to zero.  The close could
    fail if the call to rdbuf()->close() fails.  This function
    has no return value.

      void fstream::setbuf(char *p, size_t len);
    calls filebuf::setbuf(p,len).  This function has no return
    value.

      filebuf* fstream::rdbuf();
    returns a pointer to the filebuf associated with the
    stream.

Example

  #include <fstream.h>

  // Example using an fstream (Input/Output File-based stream)
```

```
int main(void)
{
    // Declare an fstream object called "mystream" and initialize
    // it to perform I/O to the file "myiofile.dat"
    fstream mystream("myiofile.dat", ios::in|ios::out);

    // Declare an unopened fstream object called "mystream2"
    fstream mystream2;

    // Declare a pointer to an fstream object
    fstream *stream_p;

    int i;

    if (!mystream)
    {
  cout << "Error opening file \"myiofile.dat\"!" << endl;
  return 20;
    }

    // Read an integer from the file attached to "mystream"
    mystream >> i;

    if(!mystream) cout << "Read from \"myiofile.dat\""
        " failed" << endl;
    else cout << "Read " << i << " from \"myiofile.dat\"" << endl;

    i = i + 1;          // Add one to the integer
    mystream.seekp(0,ios::beg); // Seek back to beginning of file

    mystream << i;       // Write the integer back

    if (!mystream) cout << "Write to \"myiofile.dat\""
         "failed" << endl;
    else cout << "Wrote " << i << " to \"myiofile.dat\"" << endl;

    // Initialize the unopened "mystream2" stream to perform I/O to
    // the file "myiofile2.dat"
    mystream2.open("myiofile2.dat", ios::app|ios::in|ios::out);

    if (!mystream2)
    {
  cout << "Error opening file \"myiofile2.dat\"!" << endl;
  return 20;
    }

    // Read an integer from "myiofile2.dat"
    mystream2 >> i;

    if (!mystream2) cout << "Read from myiofile2.dat failed"
       << endl;
    else cout << "Read " << i << " from myiofile2.dat" << endl;

    // Write the new integer.  Note that this will APPEND the new
    // integer to the old file, not replace the old file, since
    // we did not seek to the beginning of the file before writing.
    // Put a blank in to seperate the old integer from the new one.
```

```
      i = i + 1;
      mystream2 << " " << i;

      if (!mystream2) cout << "Write to myiofile2.dat failed" << endl;
      else cout << "Appended " << i << " to myiofile2.dat" << endl;

      // Allocate a new fstream using "new" and use it to perform I/O
      // to the file "myiofile3.dat"
      stream_p = new fstream("myiofile3.dat", ios::in|ios::out);

      if (!stream_p || !*stream_p)
      {
    cout << "Error opening file \"myiofile3.dat\"!" << endl;
    return 20;
      }

      *stream_p >> i;

      if (!*stream_p) cout << "Read from myiofile3.dat failed"
         << endl;
      else cout << "Read " << i << " from myiofile3.dat" << endl;

      i = i + 1;
      stream_p->seekp(0, ios::beg);  // Rewrite this one, not append
      *stream_p << i;

      if (!*stream_p) cout << "Write to file myiofile3.dat failed"
         << endl;
      else cout << "Wrote " << i << " to myiofile3.dat" << endl;

      // Free the object just allocated.  This will call the destructor
      // for the stream and therefore close the file.
      delete stream_p;

      // Destructors for the other streams will automatically be called.
      return 0;
  }
```

See Also

```
  class filebuf, class ifstream, class ofstream
```

1.346 class ifstream

class ifstream-Provide formatted file I/O

Synopsis

```
  #include <fstream.h>
  class ifstream : public istream
  {
  public:
     ifstream();
     ifstream(const char *name, int mode = ios::in);
     virtual ~ifstream();

     void open(const char *name, int mode = ios::in);
     void close();

     void setbuf(char *p, size_t len);
     filebuf* rdbuf();
  };
```

```
Description

  class ifstream is an input-only stream whose input source is a
  file.  The streambuf associated with the I/O operations is a
  filebuf (instead of a strstreambuf or stdiobuf).

Parent Class

  class ifstream inherits characteristics from class istream, which
  inherits characteristics from class ios.  See the descriptions of
  these parent classes for the details on functions and operators
  that are inherited.

Constructors

  There are two sets of constructors for class ifstream, as follows:

      ifstream::ifstream();
    creates an unopened stream of the appropriate type.  You
    can use the open() member function to open the stream
    after it is created.

      ifstream::ifstream(const char *name, int mode = ios::in);
    creates a stream of the appropriate type, named name,
    using the specified mode.  The ifstream constructor
    behaves as if ios::in was set in the mode argument,
    whether or not it was set by the caller.

    See the description of enum open_mode for information on
    the available modes.  If the open fails, the stream's
    status is reflected in its I/O state flags.  See the
    description of enum io_state for information on the the
    I/O state flags.

Destructors

  class ifstream has one destructor:

      virtual ifstream::~ifstream();
    closes the stream, if opened.

Member Functions

  The following descriptions give the purpose and return type of the
  member functions, as well as any other appropriate information.

      void ifstream::open(const char *name, int mode = ios::in);
    opens the named file using the specified mode.
    ifstream::open() behaves as if ios::in was set in the mode
    argument, whether or not it was set by the caller.

    See the description of enum open_mode for information on
    the available open modes.

    This function does not have a return value.  If an error
    occurs during the open, the ios::failbit bit is set in the
    stream's I/O state.  For example, the call to
    rdbuf()->open() may fail.

      void ifstream::close();
    close the connection between the appropriate object and
    its associated file.  Unless an error occurs, all bits in
    the object's I/O state are set to zero.  The close could
    fail if the call to rdbuf()->close() fails.  This function
    has no return value.

      void ifstream::setbuf(char *p, size_t len);
    calls filebuf::setbuf(p,len).  This function has no return
    value.

      filebuf* ifstream::rdbuf();
    returns a pointer to the filebuf associated with the
    stream.

Example

  #include <fstream.h>

  // Example using an ifstream (Input-only File-based stream)

  int main(void)
  {
      // Declare an ifstream object called "mystream" and initialize
      // it to read bytes from the file "myfile.dat"
      ifstream mystream("myfile.dat");

      // Declare an unopened ifstream object called "mystream2"
      ifstream mystream2;

      // Declare a pointer to an ifstream object
      ifstream *stream_p;

      int i;

      if (!mystream)
      {
    cout << "Error opening \"myfile.dat\"!" << endl;
    return 20;
      }

      // Read an integer from the file attached to "mystream"
      mystream >> i;

      // Print the integer to the program's standard output
      cout << "The integer in the file \"myfile.dat\" is "
```

```
     << i << endl;

      // Initialize the unopened "mystream2" stream to read from
      // the file "myfile2.dat"
      mystream2.open("myfile2.dat");

      if (!mystream2)
      {
    cout << "Error opening \"myfile2.dat\"!" << endl;
    return 20;
      }

      // Read an integer from "myfile2.dat" and print the result
      mystream2 >> i;

      cout << "The integer in the file \"myfile2.dat\" is "
     << i << endl;

      // Allocate a new ifstream using "new" and use it to read from
      // the file "myfile3.dat"
      stream_p = new ifstream("myfile3.dat");

      if (!stream_p || !*stream_p)
      {
    cout << "Error opening \"myfile3.dat\"!" << endl;
    return 20;
      }

      *stream_p >> i;
      cout << "The integer in the file \"myfile3.dat\" is "
     << i << endl;

      // Free the object just allocated.  This will call the destructor
      // for the stream and therefore close the file.
      delete stream_p;

      // Destructors for the other streams will automatically be called.
      return 0;
  }
```

```
See Also

  class filebuf, class fstream, class ofstream
```

1.347 class ios

```
class ios-Provide buffer and stream manipulation

Synopsis

  #include <iostream.h>

  class ios
  {
  public:
     /* See the enum format_state, enum io_state, enum open_mode, */
     /* and enum seek_dir descriptions for more definitions.  */

     ios(streambuf *buf);
     virtual ~ios();

     int width();
     int width(int w);

     char fill();
     char fill(char c);

     int precision();
     int precision(int i);

     static unsigned long bitalloc();
     static int xalloc();
     long & iword(int i);
     void*& pword(int i);

     streambuf* rdbuf();

     ostream* tie();
     ostream* tie(ostream *s);
  };
```

Description

The iostream.h header file declares class ios. This class and classes derived from it provide an I/O interface for inserting information into and extracting information from streambufs. All stream classes are derived from class ios. This I/O interface supports both formatted and unformatted information. This description is devoted to those operations used in stream and buffer manipulation.

Several enumerations are defined in class ios (io_state, open_mode, seek_dir, and the format flags). These enumerations are described in subsequent sections. The open_mode and seek_dir flags are not used directly by the functions in class ios, but they are used by classes derived from it. You will not normally use class ios directly, but rather one of the classes derived from it.

Parent Class

class ios is the parent of all the stream classes. It has no parent class.

Constructors

class ios defines one constructor:

```
    ios::ios(streambuf *buf);
  sets up buf as the associated streambuf.  If buf is NULL,
  the effect is undefined.
```

Destructors

class ios has one destructor:

```
    virtual ios::~ios();
  closes the stream.
```

Buffer and Stream Manipulation Functions

class ios defines several functions that provide buffer and stream manipulation capabilities. The following list describes these functions.

```
    streambuf* ios::rdbuf();
  returns a pointer to the streambuf associated with the
  ios when it was created.

    ostream* ios::tie();
  returns the ostream currently tied to the ios, if any;
  returns NULL otherwise.

    ostream* ios::tie(ostream *s);
  ties s to the ios and returns the stream previously tied
  to this stream, if any; returns NULL otherwise.
```

If the ios is tied to an ostream, then the ostream is flushed before every read or write from the ios. By default cin, cerr, and clog are tied to cout.

Formatting Functions

class ios defines several functions that use and set the format flags and variables. class ios also provides functions you can use to define and manipulate your own formatting flags, plus several built-in manipulators that allow you to set various format flags.

Format flag functions

The following list describes some of the functions that use and set the library-supplied format flags. For information on the flags(), setf(), and unsetf() functions and the format flags, see the description of enum format_state.

```
    int ios::width();
  returns an int representing the value of the current field
  width.

    int ios::width(int w);
  sets the field width to w and returns an int representing
  the previous field width value.
```

The default field width is 0. When the field width is 0, inserters insert only as many characters as necessary to represent the value. When the field width is nonzero, inserters insert at least as many characters as are necessary to fill the field width. The fill character is used to pad the value, if necessary, in this case.

Numeric inserters do not truncate their values. Therefore, if the value being inserted is wider than the field width, the entire value is inserted, regardless of the field width overrun. The field width value is a minimum constraint; you cannot specify a maximum constraint on the number of characters to be inserted.

The field width variable is reset to 0 after each insertion or extraction. In this sense, the field width serves as a parameter for insertions and extractions.

You can also use the predefined manipulator, setw, to set the field width.

```
  char ios::fill();
```
returns a char representing the current fill character.

```
  char ios::fill(char c);
```
sets the fill character to c and returns a char representing the previous value. The default fill character is a space. You can also set the fill character using the predefined manipulator, setfill.

```
  int ios::precision();
```
returns an int representing the current precision value.

```
  int ios::precision(int i);
```
sets the precision to i and returns an int representing the previous precision. Use this function to control the number of significant digits included in floating-point values. The default precision is six. You can also set the precision using the predefined manipulator, setprecision.

User-defined format flag functions

class ios includes four functions that you can use to define format flags and variables in addition to those described in the enum format_state description.

```
  static unsigned long ios::bitalloc();
```
returns an unsigned long with a single, previously unallocated, bit set. This allows you to create additional format flags. This function returns 0 if there are no more bits available. Once the bit is allocated, you can set it and clear it using the flags(), setf(), and unsetf() functions.

```
  static int ios::xalloc();
```
returns an int that represents a previously unused index into an array of words available for use as format state variables. These state variables can then be used in your derived classes.

```
  long& ios::iword(int i);
```

```
    returns a reference to the ith user-defined word.  i must
    be an index allocated by ios::xalloc().

      void*& ios::pword(int i);
    returns a reference to the ith user-defined word.  i must
    be an index allocated by ios::xalloc().  pword() is the
    same as iword() except that its return type is different.

  Refer to the C++ Language System Release 3.0 Library Manual for
  more information on defining and using user-defined format flags.

Built-in manipulators

  Manipulators take an ios&, an istream&, or an ostream& and return
  their argument.  The following built-in manipulators are useful
  with ios objects.  stream has type ios&.

      stream >> dec and stream << dec
    set the conversion base for the stream to decimal (by
    setting the ios::dec bit and clearing ios::oct and
    ios::hex).

      stream >> oct and stream << oct
    set the conversion base for the stream to octal (by
    setting the ios::oct bit and clearing ios::dec and
    ios::hex).

      stream >> hex and stream << hex
    set the conversion base for the stream to hexadecimal (by
    setting the ios::hex bit and clearing ios::dec and
    ios::oct).

      stream >> ws
    extracts whitespace characters.

      stream << endl
    inserts a newline character and flushes the stream.

      stream << ends
    inserts a null (\0) character into the stream.

      stream << flush
    flushes the given ostream object.

  In addition, you can use predefined manipulators such as setfill,
  setprecision, setiosflags, and resetiosflags to operate on ios
  objects.  For information on predefined manipulators, see the
  description of class IOMANIP.

See Also

  class iostream, class istream, class ostream
```

1.348 class ios, enum format_state

class ios, enum format_state-Provide buffer and stream formatting

Synopsis

```
  #include <iostream.h>

  class ios
  {
  public:

      /* See the class ios, enum io_state, enum open_mode, and */
      /* enum seek_dir descriptions for more definitions.      */

      enum {skipws,
      left, right, internal,
      dec, oct, hex,
      showbase, showpoint, uppercase, showpos,
      scientific, fixed,
      unitbuf, stdio
      };

      static const unsigned long basefield;
      static const unsigned long adjustfield;
      static const unsigned long floatfield;

      unsigned long flags();
      unsigned long flags(unsigned long f);

      unsigned long setf(unsigned long mask);
      unsigned long setf(unsigned long setbits, unsigned long mask);
      unsigned long unsetf(unsigned long mask);
  };
```

Description

class ios (defined in the iostream.h header file) provides a format state, which is used by the stream classes to control formatting. The format state is controlled by the format flags and class ios provides several functions to manipulate these flags. The member functions flags(), setf(), and unsetf() control the majority of formatting. These functions are described in the Formatting Functions section. Other member functions that have an effect on the format state are fill(), width(), and precision(). For information on these functions, see the description of class ios.

In addition to the predefined format flags, users can create their own user-defined format flags. For more information, see "User-defined format flag functions" in the description of class ios.

Format Flags

The following list describes each format flag in detail.

skipws

```
skips whitespace on input.  This flag applies only to
scalar extractions.  If skipws is not set, whitespace is
not skipped.

To protect against looping, zero-width fields are
considered a bad format.  Therefore, if the next character
is whitespace and skipws is not set, arithmetic extractors
signal an error.  skipws is set by default.

  padding flags
control the padding of formatted values.  There are three
of them:

    left
  left-justifies the output.

    right
  right-justifies the output.  If padding is not
  specified, right is the default value.

    internal
  causes padding to occur between the sign or base
  indicator and the value.

These padding flags are grouped together by the member
adjustfield.  To set the fill character, use the fill()
function.  To control the width of formatted items, use
the width() function.

  conversion base flags
control the conversion base of values, as follows:

    dec
  specifies decimal as the conversion base.  If a
  conversion base is not specified, dec is the
  default value.

    oct
  specifies octal as the conversion base.

    hex
  specifies hexadecimal as the conversion base.

These conversion base flags are grouped together by the
member basefield.

Although decimal is the default conversion base for
insertions (if none of these flags are set), the default
conversion base for extractions follows the C++ lexical
conventions for integral constants.  You can also use the
built-in manipulators dec, oct, and hex to control the
conversion base.  These manipulators are described in the
previous class ios description, under "Built-in
Manipulators."

  showbase
causes the base indicator to be shown in the output.  This
```

```
form of output follows the C++ lexical conventions for
integral constants.  showbase is not set by default.

  showpoint
causes the output to include any trailing zeros and
decimal points resulting from floating-point conversion.
showpoint is not set by default.

  uppercase
causes uppercase letters to be used in output of base
indicators and scientific notation.  For example, an X is
used instead of x in hexadecimal output and an E is used
instead of e in scientific notation.  uppercase is not set
by default.

  showpos
causes a + sign to be added to the decimal conversion of
positive integers.  showpos is not set by default.

  floating-point flags
control the format of floating-point conversions, as follows:

    scientific
  causes the value to be converted using scientific
  notation.  In this form, there is one digit
  preceding the decimal point and the number of
  digits after the decimal point is equal to the
  precision (set with the precision() function).
  The default precision is six.  An e (or E if
  uppercase is set) precedes the exponent.

    fixed
  causes the value to be converted to decimal
  notation. The precision of the value is controlled
  with the precision() function.  The default
  precision is six.

If neither scientific or fixed is set, the value is
converted to one or the other format, according to the
following rules:

    · If the exponent resulting from the conversion is less
      than -4 or greater than the precision, scientific
      notation is used.

    · Otherwise, fixed notation is used.

Unless showpoint is set, trailing zeros are removed from
the value, regardless of the format.  A decimal point
appears in the value only if it is followed by a digit.
These flags are grouped together by the member floatfield.
They are not set by default.

  unitbuf
causes the stream to be flushed after an insertion.
unitbuf is not set by default.
```

```
      stdio
    causes the standard C output files stdout and stderr to be
    flushed after an insertion.  stdio is not set by default.

Formatting Functions

  The following functions can be used to turn format flags on and
  off.

      unsigned long ios::flags();
    returns an unsigned long representing the current format
    flags.

      unsigned long ios::flags(unsigned long f);
    sets (turns on) all the format flags specified by f,
    unsets all format flags not specified by f, and returns an
    unsigned long representing the previous flag values.

      unsigned long ios::setf(unsigned long mask);
    sets (turns on) only those format flags that are set in
    mask and returns an unsigned long representing the
    previous values of those flags.  All other flags are left
    untouched.  You can accomplish the same task by using the
    predefined manipulator, setiosflags.

      unsigned long ios::setf(unsigned long setbits,
            unsigned long mask);
    turns on or off the flags marked by mask according to the
    corresponding values specified by setbits and returns an
    unsigned long representing the previous values of the bits
    specified by mask.  The Examples section provides an
    example of using this function.

    Using setf(0, mask) clears all the bits specified by mask.
    You can accomplish the same task by using the predefined
    manipulator resetiosflags.

      unsigned long ios::unsetf(unsigned long mask);
    clears the format flags specified by mask and returns an
    unsigned long representing the previous flag values.

Examples

  The setf() function is used to change format flags. For example,
  if you want to change the conversion base in an ios object called
  s, you could use the following expression:

      s.setf(ios::hex, ios::basefield)

  In this example, ios::basefield represents the conversion base
  bits you want to change and ios::hex is the new value.

  To set a flag that is not part of a field, use setf() with a
  single argument, as in the following example, which sets the
  skipws flag:

      s.setf(ios::skipws)
  To clear the skipws flag, use unsetf():

      s.unsetf(ios::skipws)

  As another example of using setf(), suppose you want to clear in
  your ios object s all the bits specified by the variable
  clearbits.  You could use the following expression to accomplish
  this:

      s.setf(0, clearbits)
```

1.349 class ios, enum io_state

class ios, enum io_state-Provide stream I/O state

Synopsis

```
  #include <iostream.h>

  class ios
  {
  public:
      enum io_state {goodbit = 0,
         eofbit,
         failbit,
         badbit
      };

      /* See the class ios, enum format_state, enum open_mode, and */
      /* enum seek_dir descriptions for more definitions.    */

      int rdstate();
      int eof();
      int fail();
      int bad();
      int good();
      void clear(int i = 0);

      operator void*();
      int operator !();
  };
```

Description

```
  class ios (defined in the iostream.h header file) defines io_state
  flags that represent the internal state of an ios object .  Each
  flag has a value that can be set or reset independently for an ios
  object.  goodbit is not a flag, but rather a symbolic name for the
  condition in which no flags are set.  The functions such as
  rdstate() and eof() use and manipulate the I/O state flags.
```

I/O State Flags

```
  A stream is in an unusual state (error or EOF) if any of the I/O
  state flags are set for the stream.  If none of the flags are set,
  the stream is in the normal (non-error) state.  The meaning of the
  io_state enumerators are as follows:

      goodbit
    is not a flag.  It is a symbolic name for the condition in
    which no flags are set.

      eofbit
    indicates the end of file has been encountered. If the
    stream is repositioned after eofbit is set, the bit is
    cleared.

      failbit
    indicates an error other than an I/O error, such as an
    error in formatting.  Once the failbit bit is cleared, I/O
    can usually continue.  failbit is also set if an operator
    or member function fails because no more characters can be
    extracted.

      badbit
    indicates an I/O operation failed.  It is usually
    ill-advised to continue I/O operations after this bit is
    set.

I/O State Functions

  class ios also provides several functions that use or manipulate
  the I/O state flags. In addition to the I/O state functions, class
  ios also defines two operators that allow you to check the I/O
  state of an ios object.

  The following functions use and manipulate the values of the I/O
  state flags.

      int ios::rdstate();
    returns the current I/O state.

      int ios::eof();
    returns the value of eofbit if eofbit is set; otherwise,
    returns 0.

      int ios::fail();
    returns the value of failbit if failbit is set; otherwise,
    returns 0.

      int ios::bad();
    returns the value of badbit if badbit is set; otherwise,
    returns 0.

      int ios::good();
    returns a nonzero value if no bits are set in the stream's
    I/O state; otherwise, returns 0.

      void ios::clear(int i = 0);
    sets the stream's I/O state to i. The default value for i
    is 0.  The clear() function has no return value.
  The following two operators are useful when checking the I/O state
  of an ios object.

      ios::operator void*();
    converts an ios object to a pointer.  If no bits are set
    in the stream's I/O state, this operator returns a
    non-NULL pointer value.  If failbit or badbit is set, the
    operator returns 0.

      int ios::operator !();
    converts an ios object to 0 if no bits are set in the
    stream's I/O state, or to a nonzero value if any bits are
    set in the stream's I/O state.
```

1.350 class ios, enum open_mode

```
class ios, enum open_mode-Provide buffer and stream open modes

Synopsis

  #include <iostream.h>

  class ios
  {
  public:

      /* See the class ios, enum format_state, enum io_state, */
      /* and enum seek_dir descriptions for more definitions. */

      enum open_mode {in, out, ate, app, trunc, nocreate,
          noreplace, binary
      };
  };

Description

  The open_mode enumeration is defined in iostream.h.  This
  enumeration defines a number of flags that can be used when
  creating or opening a stream to specify attributes of the stream.
  You can specify several attributes simultaneously by using the OR
  operator to combine them. For example, to specify both the out and
  binary flags, use ios::out|ios::binary.

  Only the ios::ate and ios::app flags are meaningful for string
  streams, such as strstream objects.  See the description of class
  strstream, class istrstream, and class ostrstream for information
  on how these flags are used with these classes.

  The following list describes the meaning of the open_mode flags
  for the file-oriented stream classes:

      in
    means access the file for input.
```

```
      out
    means access the file for output.  If the file already
    exists, it is truncated unless one of ios::in, ios::ate,
    or ios::app is also specified.

      ate
    means to position the put pointer to the end of the file
    when the file is opened.

      app
    means to access the file in append mode. In append mode,
    each output operation to the file causes the put pointer
    to be positioned to the end of the file before writing.

      trunc
    means to truncate the file (making it empty) when it is
    opened. ios::trunc has no effect if the file does not yet
    exist.

      nocreate
    means the open fails if the file to be opened does not
    exist.

      noreplace
    means the open fails if the file already exists.

      binary
    means to access the file in binary mode.  If ios::binary
    is not specified, the file is accessed in text mode.  See
    the description of the  fopen  function in Chapter 7 for
    information on on accessing a file in text mode (Mode A)
    and binary mode (Mode B).
```

1.351 class ios, enum seek_dir

```
class ios, enum seek_dir-Provide buffer and stream seeking

Synopsis

  #include <iostream.h>

  class ios
  {
  public:

      /* See the class ios, enum format_state, enum open_mode, */
      /* and enum open_mode descriptions for more definitions. */

      enum seek_dir {beg, cur, end};
  };

Description

  When you perform a seek on a stream, you must specify the starting
  point for the seek.  class ios (defined in the iostream.h header
  file) provides the seek_dir flags to control seeking.  The
  following list describes these flags:

      beg
    means the seek is relative to the beginning of the stream.

      cur
    means the seek is relative to the current position of the
    stream.

      end
    means the seek is relative to the end of the stream.
```

1.352 class iostream

```
class iostream-Provide bidirectional stream

Synopsis

  #include <iostream.h>

  class iostream : public ostream,
       public istream
  {
      public:
      iostream(streambuf *buf);
  };

Description

  The iostream.h header file also provides class iostream, which is
  both an istream and an ostream. class iostream includes all the
  operations of both subclasses. It adds only a constructor of its
  own. You will not normally use class iostream directly, but rather
  use one of its derived classes (fstream, strstream, or
  stdiostream).

Parent Class

  class iostream inherits characteristics from both class istream
  and class ostream.  See the descriptions of these parent classes
  for the details on functions and operators that are inherited.

Constructors

  class iostream defines one constructor:

      iostream(streambuf *buf);
    sets up buf as the associated streambuf.  If buf is NULL,
    the effect is undefined.

See Also

  class ios, class fstream, class istream, class stdiostream, class
  strstream, class ostream
```

1.353 class istream

```
class istream—Provide for stream extraction

Synopsis

  #include <iostream.h>

  class istream : virtual public ios
  {
  public:
      istream(streambuf *buf);

      virtual ~istream();

      int ipfx(int need = 0);

      istream& operator >>(char *str);
      istream& operator >>(unsigned char *str);
      istream& operator >>(signed char *str);

      istream& operator >>(char& c);
      istream& operator >>(unsigned char& c);
      istream& operator >>(signed char& c);

      istream& operator >>(short& sh);
      istream& operator >>(unsigned short& sh);

      istream& operator >>(int& i);
      istream& operator >>(unsigned int& i);

      istream& operator >>(long& l);
      istream& operator >>(unsigned long& l);

      istream& operator >>(float& f);
      istream& operator >>(double& d);
      istream& operator >>(long double& ld);

      istream& operator >>(streambuf *buf);

      istream& operator >>(istream&(*f)
        (istream&));
      istream& operator >>(ios&(*f)(ios&));

      istream& get(char *str, int len, char delim = '\n');
      istream& get(unsigned char *str, int len, char delim = '\n');
      istream& get(signed char *str, int len, char delim = '\n');

      istream& getline(char *str, int len, char delim = '\n');
      istream& getline(unsigned char *str, int len, char delim = '\n');

      istream& getline(signed char *str, int len, char delim = '\n');
      istream& get(streambuf& sb, char delim = '\n');

      istream& get(signed char& c);
      istream& get(unsigned char& c);
      istream& get(char& c);

      int get();

      istream& ignore(int n = 1, int delim = EOF);

      istream& read(char *str, int n);
      istream& read(unsigned char *str, int n);
      istream& read(signed char *str, int n);

      int gcount();
      int peek();

      istream& putback(char c);
      int sync();

      istream& seekg(streampos pos);
      istream& seekg(streamoff offset, seek_dir place);

      streampos tellg();
  };
```

Description

class istream is defined in the iostream.h header file and is the base class for those stream classes that support only input. It includes all the basic extraction functions (formatted input) on fundamental C++ types, as well as a number of unformatted input functions and several functions that enable you to move the get pointer. It also includes one manipulator. These members are described in the following sections.

You will not normally use class istream directly, but rather use one of its derived classes (ifstream or istrstream).

Parent Class

class istream inherits characteristics from class ios.

See the

description of this parent class for the details on functions and operators that are inherited.

Constructors

class istream defines one constructor:

```
      istream::istream(streambuf *buf);
```

initializes an istream and associates a streambuf with it.

Destructors

class istream has one destructor:

```
    virtual istream::~istream();
```

closes the istream.

Input Prefix Function

class istream defines an input prefix function that performs those operations that must be done before each formatting operation. This function is defined as follows:

```
    int istream::ipfx(int need = 0);
```

If any I/O state flags are set for the istream, this function returns 0 immediately. If necessary, it flushes the ios (if any) tied to this istream. Flushing is necessary if need is 0 or if there are less than need characters available for input.

If ios::skipws is set and need is 0 then this function causes any leading white space in the input to be skipped. If an error occurs during this skipping, ipfx() returns 0. If no errors have occurred, this function returns 1.

This function is called by all formatted extraction operations and should be called by user-defined extraction operators unless the first input operation used by the user-defined extraction operator is a formatted extraction. For user-defined operations, ipfx() should be called with the argument equal to 0.

Formatted Input Functions

The functions named operator >> are called extraction operators. They are formatted input functions. They each call the input prefix function, ipfx(0) and do nothing else if it returns 0. If ipfx(0) does not return 0, the formatted input functions extract leading characters from the associated streambuf according to the type of their argument and the formatting flags in the ios. They all return the address of the istream.

Errors during extraction are indicated by setting the appropriate I/O state flags for the stream, as follows:

```
    ios::failbit
```

means that the actual input characters did not match the expected input format.

```
    ios::badbit
```

means that an error occurred during extraction of characters from the streambuf.

The following list described each of the formatted input functions:

```
    istream& istream::operator >>(char *str);
    istream& istream::operator >>(unsigned char *str);
    istream& istream::operator >>(signed char *str);
```

extract characters up to the next whitespace character.

```
The terminating whitespace character is not extracted. If
width() is nonzero, these functions extract no more than
width() - 1 characters and reset width() to 0.  These
functions add a terminating null character, even if an
error occurs during extraction.

  istream& istream::operator >>(char& c);
  istream& istream::operator >>(unsigned char& c);
  istream& istream::operator >>(signed char& c);
extract a single character and store it in the argument.

  istream& istream::operator >>(short& sh);
  istream& istream::operator >>(unsigned short& sh);
  istream& istream::operator >>(int& i);
  istream& istream::operator >>(unsigned int& i);
  istream& istream::operator >>(long& l);
  istream& istream::operator >>(unsigned long& l);
extract a number and store it in the argument. There may
be a leading sign character (+ or -).  If any of ios::dec,
ios::oct, or ios::hex is set in the formatting state,
characters are extracted and converted according to the
bit that is set.  If none of these bits is set, then these
functions expect any of the following formats:

    0xhhh
    0Xhhh
    0ooo
    ddd

Extraction stops when it reaches an unacceptable
character.  The acceptable characters are

    0-7
  for octal conversion

    0-9
  for decimal conversion

    0-9, a-f, and A-F
  for hexadecimal conversion.

ios::failbit is set if no digits are found.

  istream& istream::operator >>(float& f);
  istream& istream::operator >>(double& d);
  istream& istream::operator >>(long double& ld);
extract a floating-point number and store it in the
argument. The expected input format is an optional sign,
followed by a decimal mantissa (optionally including a
decimal point), followed by an optional floating-point
exponent. The exponent may contain either an uppercase or
a lowercase E, and may have a + or a - following the E.
Extraction stops when EOF is encountered, or when a
character is read which cannot continue the previous input
in a valid manner. ios::failbit is set if there are no
digits to extract, or if the format is not correct.
```

```
      istream& istream::operator >>(streambuf *buf);
    extracts all characters from the istream and inserts them
    into the streambuf.  Extraction stops when no more
    characters can be obtained from the istream.

      istream& istream::operator >>(istream&(*f) (istream&));
      istream& istream::operator >>(ios&(*f)(ios&));
    are for support of simple manipulators.  Although these
    operators resemble an extraction in appearance, they are
    used to manipulate the stream rather than to extract
    characters from it.  The argument to either of these
    operators is a manipulator function that modifies its ios
    or istream argument in some manner.
```

Unformatted Input Functions

```
  The following functions are the unformatted input functions.  They
  each call ipfx(1) first and do nothing else if 0 is returned.

      istream& istream::get(char *str, int len, char delim = '\n');
      istream& istream::get(unsigned char *str, int len,
          char delim = '\n');
      istream& istream::get(signed char *str, int len,
          char delim = '\n');
    extract up to len - 1 characters. Extraction stops when a
    delim character is extracted, no more characters are
    available, or when len - 1 characters have been found.
    These functions store a terminating null character in the
    array. ios::failbit is set only if no characters at all
    were extracted.

      istream& istream::getline(char *str, int len,
              char delim = '\n');
      istream& istream::getline(unsigned char *str,
              int len, char delim = '\n');
      istream& istream::getline(signed char *str,
              int len, char delim = '\n');
    behave like the get() functions, except that the
    terminating delim character (if found) is extracted. A
    terminating null character is always stored in the array.

      istream& istream::get(streambuf& sb, char delim = '\n');
    extracts characters up to the next delim character or EOF
    and insert them into sb.  The delim character is not
    extracted or inserted. ios::failbit is set if an error
    occurs while inserting into sb.

      istream& istream::get(signed char& c);
      istream& istream::get(unsigned char& c);
      istream& istream::get(char& c);
    extract a single character. ios::failbit is set if no
    characters can be extracted.

      int istream::get();
    extracts a single character and returns it.  EOF is
    returned if no characters can be extracted. ios::failbit
    is never set.
```

```
      istream& istream::ignore(int n = 1, int delim = EOF);
    extracts up to the next n characters, or up to the
    next delim character.  ios::failbit is never set.

      istream& istream::read(char *str, int n);
      istream& istream::read(unsigned char *str, int n);
      istream& istream::read(signed char *str, int n);
    extract the next n characters and store them into the
    array pointed to by str.  If fewer than n characters can
    be extracted, ios::failbit is set.

Other Member Functions

  class istream includes several other functions, as described in
  the following list.

      int istream::gcount();
    returns the number of characters extracted by the last
    unformatted extraction function. Formatted extraction
    functions may change the value of this function in
    unexpected ways.

      int istream::peek();
    returns EOF if ipfx(1) returns 0 or if no characters
    remain to be extracted.  Otherwise it returns the next
    character in the stream without extracting it.

      istream& istream::putback(char c);
    does nothing if any bits are set in the stream's I/O
    state.  If no bits are set in the stream's I/O state, this
    function pushes back the character c so it will be the
    next character extracted. c must be the same as the last
    character extracted from the istream. ios::badbit is set
    if the streambuf cannot push c back.

      int istream::sync();
    calls sync() on the associated streambuf.  This function
    returns whatever the streambuf::sync() call returned.

      istream& istream::seekg(streampos pos);
      istream& istream::seekg(streamoff offset, seek_dir place);
    move the get pointer of the associated streambuf.  pos is
    a value returned by a previous call to tellg(). offset and
    place are explained in the streambuf::seekoff()
    description.

      streampos istream::tellg();
    returns the current streampos of the get pointer of the
    associated streambuf.

See Also

  class ifstream, class ios, class iostream, class istrstream,
  class ostream
```

1.354 class istrstream

```
class istrstream-Provide formatted string input

Synopsis

  #include <strstream.h>

  class istrstream : public strstreambuf,
         public istream
  {
  public:
      istrstream(char *str);
      istrstream(char *str, int size);
      ~istrstream();
      strstreambuf* rdbuf();
  };

Description

  class istrstream implements an input only stream whose source is
  an area of memory.  This class supports string (array) input by
  customizing the input operations defined in class istream.  The
  streambuf associated with class istrstream is a strstreambuf.

Parent Class

  class istrstream inherits characteristics from class istream,
  which inherits characteristics from class ios.  See the
  descriptions of these parent classes for the details on functions
  and operators that are inherited.

Constructors

  class istrstream has two constructors:

      istrstream::istrstream(char *str);
    creates a static mode istrstream such that extraction
    operations on the stream will fetch the characters of str,
    up to the terminating \0.  str must be null-terminated.
    The \0 character is not fetched. Seeks are allowed within
    the array.

      istrstream::istrstream(char *str, int size);
    creates a static mode istrstream such that extraction
    operations on the stream will fetch characters from the
    array starting at str and extending for size bytes.  Seeks
    are allowed within the array.

    For a discussion of dynamic mode and static mode streams,
    see the description of class strstreambuf, later in this
    chapter.

Destructors

  class istrstream has one destructor:
      istrstream::~istrstream();
    closes the stream.  For dynamic stream objects, closing
    means delete the array, unless it has been frozen.  For
    static stream objects, closing is meaningless.
```

Member Functions

```
  The following function is a member of class istrstream:

      strstreambuf* istrstream::rdbuf();
    returns a pointer to the strstreambuf associated with the
    stream.
```

Example

```
  #include <strstream.h>

  // Example using an istrstream (Input-only String-based stream)

  int main(void)
  {
      // Declare an istrstream object called "mystream" and initialize
      // it to read bytes from the string "123 456".
      istrstream mystream("123 456");

      // Declare a pointer to an istrstream object
      istrstream *stream_p;

      int i, j;
      double d;
      char c;

      // Read two integers from the string attached to "mystream"
      mystream >> i >> j;

      // Print the integers to the program's standard output
      cout << "The integers are " << i << " and " << j << endl;

      // Allocate a new static-mode istrstream using "new"
      stream_p = new istrstream("3.765 x");
      *stream_p >> d >> c;
      cout << "The number is " << d << " and the letter is "
     << c << endl;

      // Free the object just allocated.  This will call the destructor
      // for the stream and therefore close the file.
      delete stream_p;

      // Destructors for the other streams will automatically be called.
      return 0;
  }
```

See Also

```
  class strstream, class strstreambuf, class ostrstream
```

1.355 class ofstream

```
class ofstream-Provide formatted file I/O

Synopsis

  #include <fstream.h>

  class ofstream : public ostream
  {
  public:
      ofstream();
      ofstream(const char *name, int mode = ios::out);

      virtual ~ofstream();

      void open(const char *name, int mode = ios::out);
      void close();

      void setbuf(char *p, size_t len);
      filebuf* rdbuf();
  };

Description

  class ofstream implements an output only stream whose destination
  is a file.  The streambuf associated with the I/O operations is a
  filebuf (instead of a strstreambuf or stdiobuf).

Parent Class

  class ofstream inherits characteristics from class ostream, which
  inherits characteristics from class ios.  See the descriptions of
  these parent classes for the details on functions and operators
  that are inherited.

Constructors

  There are two sets of constructors for class ofstream, as follows:

      ofstream::ofstream();
    creates an unopened stream of the appropriate type.

     ofstream::ofstream(const char *name, int mode = ios::out);
    creates a stream of the appropriate type, named name,
    using the specified mode.  The ofstream constructor
    behaves as if ios::out was set in the mode argument,
    whether or not it was set by the caller.

    See the description of enum open_mode for information on
    the available modes.  If the open fails, the stream's
    status is reflected in its I/O state flags.  See the
    description of enum io_state for information on the the
    I/O state flags.

Destructors
```

```
  class ofstream has one destructor:

      virtual ofstream::~ofstream();
    closes the stream, if opened.
```

Member Functions

```
  The following descriptions give the purpose and return type of the
  member functions, as well as any other appropriate information.

      void ofstream::open(const char *name, int mode = ios::out);
    opens the named file using the specified mode.
    ofstream::open() behaves as if ios::out was set in the
    mode argument, whether or not it was set by the caller.
    See the description of enum open_mode for information on
    the available open modes.

    This function does not have a return value. If an error
    occurs during the open, the ios::failbit bit is set in the
    stream's I/O state.  For example, the call to
    rdbuf()->open() may fail.

      void ofstream::close();
    closes the connection between the appropriate object and
    its associated file.  Unless an error occurs, all the bits
    in the object's I/O state are set to zero.  The close
    could fail if the call to rdbuf()->close() fails.  This
    function has no return value.

      void ofstream::setbuf(char *p, size_t len);
    calls filebuf::setbuf(p,len).  This function has no return
    value.

      filebuf* ofstream::rdbuf();
    returns a pointer to the filebuf associated with the
    stream.
```

Example

```
  #include <fstream.h>

  // Example using an ofstream (Output-only File-based stream)

  int main(void)
  {
      // Declare an ofstream object called "mystream" and initialize
      // it to write bytes to the file "myofile.dat"
      ofstream mystream("myofile.dat");

      // Declare an unopened ofstream object called "mystream2"
      ofstream mystream2;

      // Declare a pointer to an ofstream object
      ofstream *stream_p;

      if (!mystream)
      {
    cout << "Error opening file \"myofile.dat\"!" << endl;
    return 20;
      }

      // Write an integer to the file attached to "mystream"
      cout << "writing \"123\" to file \"myofile.dat\"" << endl;
      mystream << 123;

      // Initialize the unopened "mystream2" stream to write to
      // the file "myofile2.dat"
      mystream2.open("myofile2.dat");

      if (!mystream2)
      {
    cout << "Error opening file \"myofile2.dat\"!" << endl;
    return 20;
      }

      // Write an integer to "myofile2.dat"
      cout << "writing \"456\" to file \"myofile2.dat\"" << endl;
      mystream2 << 456;

      // Allocate a new ofstream using "new" and use it to write to
      // the file "myofile3.dat"
      stream_p = new ofstream("myofile3.dat");

      if (!stream_p || !*stream_p)
      {
    cout << "Error opening file \"myofile3.dat\"!" << endl;
    return 20;
      }
      cout << "Writing \"789\" to file \ "myofile3.dat\"" << endl;
      *stream_p << 789;

      // Free the object just allocated.  This will call the destructor
      // for the stream and therefore close the file.
      delete stream_p;

      // Destructors for the other streams will automatically be called.
      return 0;
  }
```

See Also

```
  class filebuf, class fstream, class ifstream, class ios,
  class ostream
```

1.356 class ostream

class ostream-Provide for stream insertion

Synopsis

```
  #include <iostream.h>
```

```
class ostream : virtual public ios
{
public:
    ostream(streambuf *buf);
    virtual ~ostream();

    int opfx();

    void osfx();

    ostream& operator <<(char c);
    ostream& operator <<(signed char c);
    ostream& operator <<(unsigned char c);

    ostream& operator <<(const char *str);
    ostream& operator <<(const unsigned char *str);
    ostream& operator <<(const signed char *str);

    ostream& operator <<(short sh);
    ostream& operator <<(unsigned short sh);

    ostream& operator <<(int i);
    ostream& operator <<(unsigned int i);

    ostream& operator <<(long l);
    ostream& operator <<(unsigned long l);

    ostream& operator <<(float f);
    ostream& operator <<(double d);

    ostream& operator <<(void *vp);

    ostream& operator <<(streambuf *buf);

    ostream& operator <<(ostream&(*f)
           (ostream&));
    ostream& operator <<(ios&(*f)(ios&));

    ostream& put(char c);

    ostream& write(const char *str, int n);
    ostream& write(const signed char *str, int n);
    ostream& write(const unsigned char *str, int n);

    ostream& flush();

    streampos tellp();

    ostream& seekp(streampos pos);
    ostream& seekp(streamoff offset, seek_dir place);
};

ostream& endl(ostream&);
ostream& ends(ostream&);
ostream& flush(ostream&);
```

Description

class ostream is declared in the iostream.h header file and is the base class for those classes that support only output. It includes all the basic insertion operators (formatted output) on fundamental C++ types, as well as a number of unformatted output functions and functions designed to change the stream position. In addition, some output manipulators are defined for use with this class.

You will not normally use class ostream directly, but rather one of its derived classes, ofstream or ostrstream.

Parent Class

class ostream inherits characteristics from class ios. See the description of this parent class for the details on functions and operators that are inherited.

Constructors

class ostream defines one constructor:

```
ostream::ostream(streambuf *buf);
```

initializes an ostream and associates a streambuf with it.

Destructors

class ostream defines one destructor:

```
virtual ostream::~ostream();
```

closes the ostream.

Prefix and Suffix Output Functions

Certain operations are defined to happen either before or after formatted output through an ostream. The prefix operations are done by ostream::opfx() and the suffix operations are done by ostream::osfx().

```
int ostream::opfx();
```

performs prefix operations for an ostream. This function returns 0 and does nothing else if any bits in the stream’s I/O state are set; it returns 1 otherwise. If the ostream is tied to another stream, the other stream is flushed. For more information on the tie() and flush() functions, see "Frequently Used Stream Class Member Functions."

By convention, opfx() is called before any formatted output operation on a stream. If it returns 0 (meaning one or more bits are set in the stream’s I/O state) the output operation is not performed. Each of the built-in inserters follows this convention. User-defined formatted output functions should also follow this convention by calling this function and checking the return code before doing any output.

```
    void ostream::osfx();
  performs suffix operations on the stream. If ios::unitbuf
  is set, this ostream is flushed.  If ios::stdio is set,
  cout and cerr are flushed.  This function should be called
  at the end of any formatted output function that does
  unformatted output on the ostream.  It need not be called
  if the last output operation on the ostream was formatted.
```

Formatted Output Functions

The functions named operator << are called inserters (because they insert values into the output stream). All inserters are formatted output operations and as such follow the formatted output conventions mentioned previously.

All of the inserters do the following: First, they call opfx() and if it returns 0, they do nothing else. If there is no error, they then convert the input argument to a converted value (a sequence of characters), based on the argument's type and value and on the formatting flags set for the stream. The rules for construction of the converted value are given here for each inserter.

Once a converted value has been determined, it is copied, possibly with the addition of fill characters, to an output field. The characters of the output field are then inserted into the stream's buffer. The ios::width() function for the stream determines the minimum number of characters in the output field. If the converted value had fewer characters, fill characters (defined by the value of ios::fill() for the stream) are added to pad out the field. The placement of fill characters is as follows:

```
    ios::right
  places the converted value in the rightmost portion of the
  field (leading padding).

    ios::left
  places the converted value in the leftmost portion of the
  field (trailing padding).

    ios::internal
  places the sign and base indicators of the converted value
  in the leftmost portion of the field and the remainder in
  the rightmost portion (internal padding).
```

Truncation cannot occur when copying the converted value to an output field, regardless of the value of width().

Once the converted value is constructed and the field padded to be at least ios::width() characters wide, ios::width() is reset to 0 and osfx() is called. All inserters indicate errors by setting I/O state flags in the ostream, as necessary. Inserters always return a reference to their ostream argument.

The following list describes the formatted output functions.

```
    ostream& ostream::operator <<(char c);
```

```
      ostream& ostream::operator <<(signed char c);
      ostream& ostream::operator <<(unsigned char c);
    convert the argument to the char c.

      ostream& ostream::operator <<(const char *str);
      ostream& ostream::operator <<(const unsigned char *str)
      ostream& ostream::operator <<(const signed char *str);
    convert the argument to a sequence of chars, up to but not
    including a \0 character, pointed to by const char* str.

      ostream& ostream::operator <<(short sh);
      ostream& ostream::operator <<(unsigned short sh);
      ostream& ostream::operator <<(int i);
      ostream& ostream::operator <<(unsigned int i);
      ostream& ostream::operator <<(long l);
      ostream& ostream::operator <<(unsigned long l);
    convert the value of the argument to a sequence of digits,
    preceded by a leading minus sign (-) if the argument is
    negative.

  If the following format flags within the ostream are set, they
  affect the converted value as follows:

      ios::showpos
    causes a leading plus sign (+) to be included in the
    converted value if the value is positive.

      ios::dec, ios::oct, and ios::hex
    determine the base used for the converted value.

      ios::showbase
    causes the converted value to indicate the appropriate
    base as follows:

        decimal
      makes no change to the converted value.

        octal
      prefixes the converted value with a single 0
      digit.  If the value is 0 there is only one zero
      digit.

        hexadecimal
      prefixes the converted value with 0x.  If
      ios::uppercase is set, a leading 0X is used
      instead.

    If both a sign representation (+ or -) and a base
    representation appear in the converted value, the sign
    appears first.

      ostream& ostream::operator <<(float f);
      ostream& ostream::operator <<(double d);
    converts the argument to a character representation of its
    value in one of two formats:

        · fixed notation ("\pm ddd.ddd")
        · scientific notation ("\pm d.ddde\pm dd").

    These formats are described in detail in the enum
    format_state description.  The format of the converted
    value is affected by the settings of the following format
    flags:

        ios::fixed or ios::scientific
      determines the overall representation format.  If
      neither is set then the overall format is
      scientific if the exponent is less then -4 or
      greater than the precision. Fixed notation is
      chosen otherwise.

        ios::showpoint
      causes the decimal point to be shown, followed by
      at least one digit.  If showpoint is not set and
      all digits after the decimal point are zero, these
      digits and the decimal point are dropped.

        ios::uppercase
      causes the e in scientific notation to be E and
      the x in hexadecimal notation to be X.

        ios::showpos
      causes a leading plus sign (+) to be output for
      positive values.

      ostream& ostream::operator <<(void *vp);
    converts the value of the pointer vp to an unsigned long
    and represents it as if ios::hex and ios::showbase were
    set.

      ostream& ostream::operator <<(streambuf *buf);
    fetches all the characters in buf and inserts them into
    the output stream, provided no bits are set in buf's I/O
    state.  No padding is done.  If any bits are set in buf's
    I/O state, this function returns immediately.

      ostream& ostream::operator <<(ostream&(*f) (ostream&));
      ostream& ostream::operator <<(ios&(*f)(ios&));
    are for support of simple manipulators.  Although these
    operators resemble an insertion in appearance, they are
    used to manipulate the stream rather than to insert
    characters into it.  The argument to either of these
    operators is a manipulator function that modifies its ios
    or ostream argument in some manner.

Unformatted Output Functions

  The following functions are for support of unformatted output to a
  stream.  Because they are unformatted operations they do not call
  opfx() and osfx().  However, these functions check whether any I/O
  state flags are set for the ostream and if so, take no additional
  action.  All inserters indicate errors by setting I/O state flags
  in the ostream.  Each of these functions returns a reference to
  its argument ostream.

      ostream& ostream::put(char c);
    inserts its argument into the stream.

      ostream& ostream::write(const char *str, int n);
      ostream& ostream::write(const signed char *str, int n);
      ostream& ostream::write(const unsigned char *str, int n);
    insert n characters starting at str into the stream.  The
    characters are treated as plain chars independent of their
    actual type.  The null character is treated the same as
    any other character.

Other Member Functions

  The following functions are also members of class ostream.

      ostream& ostream::flush();
    calls rdbuf()->sync().  For more information, see the
    description of streambuf::sync().

      streampos ostream::tellp();
    returns the stream's current put pointer position.  For
    more information, see the descriptions of
    streambuf::seekoff() and streambuf::seekpos().

      ostream& ostream::seekp(streampos pos);
      ostream& ostream::seekp(streamoff offset, seek_dir place);
    reposition the stream's put pointer.  For more
    information, see the descriptions of streambuf::seekoff()
    and streambuf::seekpos().

Manipulators

  The following functions are called manipulators.  They are
  intended to be used with the inserters to manipulate the stream in
  specified ways.  These manipulators do nothing if any of the
  stream's I/O state flags are set.  They signal errors by setting
  flags in the stream's I/O state. They each return their argument.

      ostream& endl(ostream&)
    inserts a \n character into the stream.  For example:

        #include <iostream.h>

        float mynum=3.2;

        // Writes "mynum is:" & the value of mynum on one line.
        cout << "mynum is: " << mynum << endl;

      ostream& ends(ostream&)
    inserts a \0 character into the stream.  The following
    example uses a strstream:

        #include <iostream.h>
        strstream mystream;
        float mynum=3.2;
        // Writes mynum to mystream.
        mystream << "mynum is: " << mynum << ends;


      ostream& flush(ostream&)
    calls ostream.flush().
```

See Also

  class ios, class iostream, class istream

1.357 class ostrstream

class ostrstream-Provide formatted string I/O

Synopsis

```
  #include <strstream.h>

  class ostrstream : public strstreambuf,
         public ostream
  {
  public:
      ostrstream(char *str, int size, int mode = ios::out);
      ostrstream();
      ~ostrstream();

      char* str();
      int pcount();
      strstreambuf* rdbuf();
  };
```

Description

  class ostrstream implements an output only stream whose
  destination is an area of memory. This class supports string
  (array) output by customizing the output operations defined in
  class ostream. The streambuf associated with class ostrstream is
  a strstreambuf.

Parent Class

  class ostrstream inherits characteristics from class ostream and
  class ios. See the descriptions of these parent classes for the
  details on functions and operators that are inherited.

Constructors

  class ostrstream has two constructors:

```
      ostrstream::ostrstream(char *str, int size,
           int mode = ios::out);
    creates a static mode ostrstream referencing an area of
    size bytes starting at the character pointed to by str.
    The get pointer is positioned at the beginning of the
    array.  The put pointer is also positioned at the
    beginning of the array unless either the ios::ate or
    ios::app bit is set in mode; if either of these bits is
    set, the put pointer is positioned at the space that
    contains the first null character.  Seeks are allowed
    anywhere within the array.

      ostrstream::ostrstream();
    creates a dynamic mode ostrstream.  This involves
    dynamically allocating space to hold stored characters.
    Seeks are not allowed.

  For a discussion of dynamic mode and static mode streams, see the
  description of class strstreambuf.

Destructors

  class ostrstream has one destructor:

      ostrstream::~ostrstream();
    closes the stream.  For dynamic stream objects, closing
    means delete the array, unless it has been frozen.  For
    static stream objects, closing is meaningless.

Member Functions

  The following functions are members of class ostrstream.

      char* ostrstream::str();
    calls str() on the associated streambuf.  This function
    returns whatever the streambuf::str() call returned.

      int ostrstream::pcount();
    returns the number of stored bytes.

      strstreambuf* ostrstream::rdbuf();
    returns a pointer to the strstreambuf associated with the
    stream.

Example

  #include <strstream.h>

  // Example using an ostrstream (Output-only String-based stream)

  int main(void)
  {
      // Declare an ostrstream object called "mystream"
      ostrstream mystream;

      // Write two integers to the string attached to "mystream"
      mystream << 123 << 456;

      // Obtain the contents of mystream and send them to stdout
      cout << "mystream contains: " << mystream.str();
      // The destructor for the stream will automatically be called.
     return 0;
  }
```

See Also

  class istrstream, class strstream, class strstreambuf

1.358 class stdiostream

class stdiostream-Provide formatted I/O in programs using C and C++

Synopsis

```
  #include <stdiostream.h>

  class stdiostream : public iostream
  {
  public:
      stdiostream(FILE *file);

      FILE* stdiofile();

      stdiobuf* rdbuf();
  };
```

Description

  class stdiostream is declared in the stdiostream.h header file. It provides iostream access to an external file accessed by C functions using the ANSI standard I/O interfaces declared in stdio.h. Use of class stdiostream enables a program to use stdio output and C++ iostream output in the same output file. Similarly, class stdiostream enables your program to use stdio input and C++ iostream input to process the same input file.

Parent Class

  class stdiostream inherits characteristics from class iostream, which in turn inherits characteristics from class istream, class ostream, and class ios. See the descriptions of these parent classes for the details on functions and operators that are inherited.

Constructors

  class stdiostream has one constructor:

```
      stdiostream::stdiostream(FILE *file);
```
    creates a stream from the open FILE* file. The constructor assumes that the file is open.

Member Functions

  The following descriptions give the purpose and return type of the

```
  member functions, as well as any other appropriate information.

      stdiostream::FILE* stdiofile();
    returns the FILE* associated with this stream.

      stdiobuf* stdiostream::rdbuf();
    returns a pointer to the stdiobuf associated with the stream.

See Also

  class stdiobuf
```

1.359 class streampos

```
class streampos-Mark a stream location

Synopsis

  #include <iostream.h>

  class streampos
  {
  public:
      streampos();
      streampos(long n);
      operator long();
      fpos_t* fpos();
  };

Description

  class streampos is declared in the iostream.h header file and is
  used to record or specify a position in a stream.  This class is
  for use only with streams that support seeking.  class streampos
  is used as a return value for the various stream positioning
  functions defined in the different buffering classes.

  Most streampos values are similar to C fpos_t values; that is,
  they record file position values in a way private to the
  implementation.  Because these values are probably not useful for
  user-defined stream classes, it is also possible to create
  streampos objects with integral values.  streampos objects with
  integral values are probably not useful for positioning fstream or
  stdiostream objects.

Constructors

  This class defines two constructors:

      streampos::streampos();
    creates a streampos object with an unknown value.

      streampos::streampos(long n);
    creates a streampos object from the value n.  All kinds of
    stream buffers support the following values of n:
        streampos(0)
      indicates the beginning of the stream.

        streampos(EOF)
      indicates the end of the stream.

    strstream objects created from other values of n are not
    useful for positioning fstream or stdiostream objects.

Member Functions

  This class defines two member functions:

      streampos::operator long();
    reduces the value of the streampos to a long integer.  The
    value of (long)(streampos(long_val)) is defined to be
    equal to long_val.  The result of converting a streampos
    not constructed from a long is undefined.

      fpos_t* streampos::fpos();
    returns a pointer to an fpos_t contained in the streampos.
    This fpos_t contains a valid value only if this streampos
    was returned from a call to seekoff() or seekpos() on a
    filebuf or stdiobuf object.
```

1.360 class strstream

```
class strstream-Provide formatted string I/O

Synopsis

  #include <strstream.h>

  class strstream : public strstreambuf,
        public iostream
  {
  public:
      strstream(char *str, int size, int mode);
      strstream();
      ~strstream();

      char* str();
      int pcount();
      strstreambuf* rdbuf();
  };

Description

  class strstream and its associated classes class istrstream and
  class ostrstream are declared in the strstream.h header file.
  These classes support string (array) I/O by customizing the I/O
  operations defined in the base classes istream, ostream, and
  iostream.
```

Parent Class

class strstream inherits characteristics from class iostream, which inherits characteristics from class ios. See the descriptions of these parent classes for the details on functions and operators that are inherited.

Constructors

class strstream has two constructors.

```
strstream::strstream(char *str, int size, int mode);
```

creates a static mode strstream referencing an area of size bytes starting at the character pointed to by str. The get pointer is positioned at the beginning of the array. The put pointer is also positioned at the beginning of the array unless either the ios::ate or ios::app bit is set in mode; if either of these bits is set, the put pointer is positioned at the space that contains the first null character.

```
strstream::strstream();
```

creates a dynamic mode strstream. This involves allocating space to hold stored characters. Seeks are not allowed. The get pointer is positioned at the beginning of the array.

For a discussion of dynamic mode and static mode streams, see the description of class strstreambuf.

Destructors

class strstream has one destructor:

```
strstream::~strstream();
```

closes the stream. For dynamic stream objects, closing means delete the array, unless it has been frozen. For static stream objects, closing is meaningless.

Member Functions

The following functions are members of class strstream:

```
char* strstream::str();
```

calls str() on the associated streambuf. This function returns whatever the streambuf::str() call returned.

```
int strstream::pcount();
```

returns the number of stored bytes.

```
strstreambuf* strstream::rdbuf();
```

returns a pointer to the strstreambuf associated with the stream.

Example

```
// This example creates a strstream, inserts a string and a number,
// then extracts them again, writing the contents of mystream to
// cout.

strstream mystream;   // dynamic mode strstream
float mynum=3.2;
float num2;

// Write mynum to mystream.
mystream << mynum << ends;

// Extract the contents of mystream and store them in num2.
mystream >> num2;

// Get the string from mystream and write it to cout.
cout << mystream.str();
```

See Also

  class istrstream, class ostrstream, class strstreambuf

1.361 class filebuf

class filebuf-Provide file I/O

Synopsis

```
  #include <fstream.h>

  class filebuf : public streambuf
  {
  public:
      filebuf();

      virtual ~filebuf();

      int is_open();
      filebuf* open(const char *name, int mode);
      filebuf* close();
      virtual streampos seekoff(streamoff offset, seek_dir place,
              int mode = ios::in|ios::out);
      virtual streampos seekpos(streampos pos,
              int mode = ios::in|ios::out);
      virtual streambuf* setbuf(char *p, size_t len);
      virtual int sync();
  };
```

Description

  The fstream.h header file defines class filebuf. filebuf objects
  represent the lowest level of file I/O that is standard C++. They
  provide a specialized form of streambuf that uses a file as the
  source or destination for characters. Input corresponds to file
  reads and output corresponds to file writes. For filebuf objects,
  the get and put pointers are tied together. That is, if you move
  one, you move the other. If the file has a format that allows

seeks, a filebuf allows seeks. filebuf I/O guarantees at least four characters of putback. You do not need to perform any special action between reads and writes (in contrast to standard C I/O, where such seeks are required).

When a filebuf is connected to a file, the filebuf is said to be open. There is no default open mode, so you must always specify the open mode when you create a filebuf.

filebuf objects may directly access the native I/O facilities of the system on which they are implemented. For Version 6.50 of the SAS/C Development System, filebuf objects are implemented in terms of C FILE*s. This may be changed in later versions of this library and no assumptions should be made of this particular implementation.

Parent Class

class filebuf inherits characteristics from class streambuf. See the description of this parent class for the details on functions and operators that are inherited.

Constructors

class filebuf defines one constructor:

```
filebuf::filebuf();
```
creates an unopened file.

Destructors

class filebuf has one destructor:

```
virtual filebuf::~filebuf();
```
closes the file, if opened.

Non-Virtual Member Functions

The following non-virtual functions are defined in class filebuf. The virtual functions are described later in this section.

```
int filebuf::is_open();
```
returns a nonzero value if the filebuf is connected to an open file; returns 0 otherwise.

```
filebuf* filebuf::open(const char *name, int mode);
```
opens a file named name and connects the filebuf to it. If the open is successful, open() returns a pointer to the filebuf. If an error occurs during the open (for example, if the file is already open), open() returns 0. See the description of enum open_mode for information about the mode argument.

```
filebuf* filebuf::close();
```
causes any outstanding output to be flushed, then closes the file and disconnects the filebuf from it (even if errors occur). Also, the filebuf’s I/O state is cleared.

```
    If the close is successful, close() returns a pointer to
    the filebuf.  If an error occurs during the close, close()
    returns 0.

Virtual Member Functions

  The following functions override their base class definitions (in
  class streambuf).

      virtual streampos filebuf::seekoff(streamoff offset,
                 seek_dir place,
                 int mode = ios::in|ios::out);
    sets the get and put pointers to a new position, as
    indicated by place and offset.  (Descriptions of offset
    and place are contained in the streambuf::seekoff()
    description.)

    This function returns the new position, or streampos(EOF)
    if an error occurs (for example, the file may not support
    seeking, or you may have requested a seek to a position
    preceding the beginning of the file). The position of the
    file after an error is undefined. Some files support
    seeking in full and some impose lesser or greater
    restrictions on seeking.  seekoff() corresponds to the C
     fseek  function and seekpos() corresponds to the C  fsetpos
    function.  Rules for these similar C functions are in
    Chapter 7.

    For filebuf objects, the get and put pointers are the same
    (moving either one moves the other). Therefore, you do not
    have to use the last argument, mode.

      virtual streampos filebuf::seekpos(streampos pos,
                 int mode = ios::in|ios::out);
    sets the get and/or put pointers to a new position, as
    indicated by streampos.  This function returns the new
    position, or seekpos(EOF) if an error occurs.  For filebuf
    objects, the get and put pointers are the same (moving
    either one moves the other).  Because of this, you do not
    have to use the last argument, mode.

      virtual streambuf* filebuf::setbuf(char *p, size_t len);
    offers the character array starting at p and containing
    len bytes as a buffer for use by the filebuf.  If p is
    null or len is less than or equal to 0, the filebuf is
    unbuffered. (However, buffering by the SAS/C Library and
    the operating system may still take place.)  This function
    must be called before any I/O is requested for this
    filebuf and can be called only once for the filebuf.
    Under normal conditions, setbuf() returns a pointer to the
    filebuf.

    If this function is called after I/O has been requested
    for the filebuf, this function does nothing and returns
    NULL.  If this function is called more than once,
    subsequent calls for the filebuf do nothing except return
    NULL.  This function does not affect the I/O state of the
    filebuf.

      virtual int filebuf::sync();
    tries to force the state of the get or put pointer of the
    filebuf to be synchronized with the state of the file it
    is associated with. This attempt at synchronization may
    result in the following:

        · characters being written to the file, if some have
          been buffered for output.  All characters may not be
          written immediately due to additional buffering
          performed by the operating system.

        · an attempt to seek the file, if characters have been
          read and buffered for input.

    This function usually returns 0; if synchronization is not
    possible it returns EOF.
```

See Also

  class fstream, class ifstream, class ofstream

1.362 class stdiobuf

class stdiobuf-Provide I/O in a program using C and C++

Synopsis

```
  #include <stdiostream.h>

  class stdiobuf : public streambuf
  {
  public:
      stdiobuf(FILE *file);

      virtual ~stdiobuf();

      int is_open();

      FILE* stdiofile();

      streampos seekoff(streamoff offset, seek_dir place,
            int mode = ios::in|ios::out);
      streampos seekpos(streampos pos, int mode = ios::in|ios::out);
      virtual int sync();
  };
```

Description

  The stdiostream.h header file declares class stdiobuf. stdiobufs
  are intended to be an interface to ANSI C style FILE*s on those
  systems that provide FILE*s. Calls to stdiobuf member functions
  are mapped directly to calls to ANSI C stdio functions.

```
  Because stdiobuf objects provide no buffering other than that
  provided by the C stdio functions, any changes to file attributes
  or contents made via a stdiobuf are reflected immediately in the
  stdio data structures. This includes file positioning using
  seekoff() or seekpos().  For stdiobuf objects, the get and put
  pointers are tied together.  That is, if you move one, you move
  the other.

  Unless you are mixing streambuf and stdio access to the same file,
  you should use class filebuf instead of class stdiobuf.  Use of
  filebuf objects may improve performance.

Parent Class

  class stdiobuf inherits characteristics from class streambuf.  See
  the description of this parent class for the details on functions
  and operators that are inherited.

Constructors

  class stdiobuf defines one constructor:

      stdiobuf::stdiobuf(FILE *file);
    creates a stdiobuf object associated with an open FILE*.

Destructors

  class stdiobuf defines one destructor:

      virtual stdiobuf::~stdiobuf();
    closes the associated FILE*, if opened.

Non-Virtual Member Functions

  The following descriptions detail the non-virtual member functions
  for class stdiobuf.  The redefined virtual functions are described
  later in this section.

      int stdiobuf::is_open();
    returns a nonzero value if the stdiobuf is connected to an
    open file; returns 0 otherwise.

      FILE* stdiofile();
    returns the associated FILE*.

Virtual Member Functions

  The following functions override their base class definitions (in
  class streambuf).

      streampos stdiobuf::seekoff(streamoff offset,
          seek_dir place,
          int mode = ios::in|ios::out);
    moves the get and/or put pointers of the streambuf.  place
    can be one of the following:

        ios::beg
      indicates the start of file.

        ios::cur
      indicates the current get and put position.

        ios::end
      indicates the end of file.

    offset is a positive or negative integer position relative
    to place.

    mode can be one of the following:

        ios::in
      moves the get pointer.

        ios::out
      moves the put pointer.

        ios::in|ios::out
      moves both pointers.

    Whether seekoff() works for a stdiobuf depends on the
    characteristics of the associated FILE*.

      streampos stdiobuf::seekpos(streampos pos,
          int mode = ios::in|ios::out);
    moves the get and/or put pointers of the streambuf.

    pos must be a value returned by a previous call to
    seekoff().

    mode can be one of the following:

        ios::in
      moves the get pointer.

        ios::out
      moves the put pointer.

        ios::in|ios::out
      moves both pointers.

    Whether seekpos() works for a stdiobuf depends on the
    characteristics of the associated FILE*.

    Some stream buffers do not support seeking.  For those
    stream buffers, seekpos() returns streampos(EOF) to
    indicate an error occurred.  See the documentation for
    specific stream buffer classes (such as filebuf) for more
    information on what kinds of seeking are allowed.

      virtual int stdiobuf::sync();
    tries to force the state of the get or put pointer of the
    stdiobuf to be synchronized with the state of the
    associated file.  This attempt at synchronization may
    result in the following:
        · characters being written to the file, if some have
          been buffered for output.  All characters may not be
          written immediately due to additional buffering
          performed by the operating system.

        · an attempt to seek the file, if characters have been
          read and buffered for input.

    This function usually returns 0; if synchronization is not
    possible it returns EOF.
```

See Also

```
  class stdiostream
```

1.363 class streambuf

class streambuf-Provide base class for all stream buffers

Synopsis

```
  #include <iostream.h>

  class streambuf
  {
  public:
      int in_avail();
      int out_waiting();

      int sbumpc();
      int sgetc();
      int sgetn(char *s, int n);
      int snextc();
      void stossc();

      int sputbackc(char c);
      int sputc(int c);
      int sputn(const char *s, int n);

      virtual int sync();
      virtual streampos seekoff(streamoff offset, seek_dir place,
              int mode = ios::in|ios::out);
      virtual streampos seekpos(streampos pos,
              int mode = ios::in|ios::out);

      virtual streambuf* setbuf(char *p, size_t len);
  };
```

Description

```
  class streambuf is declared in the iostream.h header file and is
  the base class for all stream buffers. Stream buffers manage the
  flow of characters between the program and the ultimate sources or
  consumers of characters, such as external files. The streambuf
```

class defines behavior common to all stream buffers. More specialized classes can be derived from class streambuf to implement appropriate buffering strategies for particular stream types. For example, filebufs implement buffering suitable for file input or output and strstreambufs implement buffering suitable for transfer of data from strings in memory. A streambuf is almost never used directly (classes derived from it are used instead), but more often acts as an interface specification for derived classes.

The functions defined by the streambuf interface are divided into two groups: non-virtual functions and virtual functions. These sets of functions are described separately.

Constructors and Destructors

class streambuf defines two constructors and one destructor. All these functions are protected. This ensures that class streambuf is used only as a base class for derived classes.

Non-Virtual Member Functions

The following list describes the non-virtual streambuf interface.

```
int streambuf::in_avail();
```
returns the number of characters that have been buffered for input. That is, the number of characters that have been read from the ultimate source of the input but have not been extracted from the streambuf. Generally, this information is useful only for classes derived from class streambuf.

```
int streambuf::out_waiting();
```
returns the number of characters that have been buffered for output. That is, the number of characters that have been inserted into the streambuf, but have not been delivered to its ultimate destination. Generally, this information is useful only for classes derived from class streambuf.

```
int streambuf::sbumpc();
```
advances the get pointer one character and returns the character preceding the advanced pointer. If the get pointer is at the end of the stream, the get pointer is not moved and EOF is returned.

```
int streambuf::sgetc();
```
returns the character following the get pointer. This function does not move the get pointer. If the get pointer is at the end of the stream, this function returns EOF.

```
int streambuf::sgetn(char *s, int n);
```
extracts the next n characters from the stream into s and positions the get pointer after the last extracted character. If there are less than n characters between the get pointer and the end of the stream, those

```
    characters are extracted into s and the get pointer is
    moved to the end of the stream. This function returns the
    number of characters extracted into s.

      int streambuf::snextc();
    advances the get pointer one character and returns the
    character after the advanced pointer.  If the get pointer
    is at the end of the stream, the get pointer is not moved
    and EOF is returned.

      void streambuf::stossc();
    advances the get pointer one character.

      int streambuf::sputbackc(char c);
    backs the get pointer up one character, returning c.  c
    must be the character that the get pointer is moved over;
    if it is not, the effects of this function are undefined.
    For example, if the get pointer is at the start of the
    stream and you call this function, the effect is
    undefined.

    Also, each class derived from streambuf may impose a limit
    on the number of characters that can be moved over by
    sputbackc().  If you exceed this limit, this function
    returns EOF.  For some classes, you cannot back up over
    any characters.

      int streambuf::sputc(int c);
    stores c in the position following the put pointer,
    replacing any pre-existing character, and then advances
    the put pointer one position.  This function returns c if
    the operation is successful, or EOF if an error occurs.

      int streambuf::sputn(const char *s, int n);
    stores after the put pointer the first n characters
    addressed by s, replacing any pre-existing characters in
    those positions, and then advances the put pointer past
    the last character stored. This function returns the
    number of characters successfully stored.

Virtual Member Functions

  The following virtual functions are members of class streambuf.
  These functions can be redefined in derived classes (both those
  supplied by the library and those you define yourself) to
  customize the behavior of streambuf objects.

      virtual int streambuf::sync();
    tries to force the state of the get or put pointer of the
    streambuf to be synchronized with the state of the sink or
    source it is associated with.  This function returns 0 if
    successful, or EOF if an error occurs.

      virtual streampos streambuf::seekoff(streamoff offset,
             seek_dir place,
                 int mode = ios::in|ios::out);
    moves the get and/or put pointers of the streambuf.  place
```

```
can be one of the following:

    ios::beg
  indicates the start of file.

    ios::cur
  indicates the current get or put position.

    ios::end
  indicates the end of file.

offset is a positive or negative integer position relative
to place.

mode can be one of the following:

    ios::in
  moves the get pointer.

    ios::out
  moves the put pointer.

    ios::in|ios::out
  moves both pointers.  This is the default value.

Some kinds of seeking are not supported for certain stream
buffers.  If a particular stream buffer does not support a
the seeking specified, this function returns
streampos(EOF) to indicate an error occurred.  See the
documentation for specific stream buffer classes (such as
filebuf) for more information on what kinds of seeking are
allowed.

  virtual streampos streambuf::seekpos(streampos pos,
             int mode = ios::in|ios::out);
moves the get and/or put pointers of the streambuf.

pos must be a value returned by a previous call to
seekoff().

mode can be one of the following:

    ios::in
  moves the get pointer.

    ios::out
  moves the put pointer.

    ios::in|ios::out
  moves both pointers. This is the default value.

Some stream buffers do not support seeking.  For those
stream buffers, seekpos() returns streampos(EOF) to
indicate an error occurred.  See the documentation for
particular stream buffer classes (such as filebuf) for
more information on what kinds of seeking are allowed.
```

```
      virtual streambuf* streambuf::setbuf(char *p, size_t len);
    allocates a buffer area to be used for buffering within
    the streambuf.
```

1.364 class strstreambuf

```
class strstreambuf-Provide string I/O

Synopsis

  #include <strstream.h>

  class strstreambuf : public streambuf
  {
  public:
      strstreambuf();
      strstreambuf(int len);
      strstreambuf(void*(*a)(long), void (*f)(void*));
      strstreambuf(char *b, int size, char *pstart = 0);

      ~strstreambuf();

      void freeze(int n = 1);
      char* str();

      virtual streambuf* setbuf(char *p, size_t len);
      int sync();
      virtual streampos seekoff(streamoff offset, seek_dir place,
              int mode = ios::in|ios::out);
      virtual streampos seekpos(streampos pos,
              int mode = ios::in|ios::out);
  };

Description

  The header file strstream.h declares class strstreambuf, which
  specializes class streambuf to provide for I/O using arrays of
  char (strings).

  For strstreambuf objects, the get and put pointers are separate.
  That is, if you move one of these pointers you do not necessarily
  move the other.

  strstreambuf objects are created in one of two different modes,
  dynamic mode or static (fixed) mode.  Once a strstreambuf is
  created it does not change modes.  The following list explains the
  difference between fixed and dynamic mode:

      dynamic mode
    means the strstreambuf does not have a fixed size and
    grows as needed.  When the array associated with a dynamic
    mode strstreambuf is filled, the strstreambuf
    automatically allocates a larger array, copies the old
    smaller array to the larger, and frees the smaller array.
    The functions used to handle allocating and freeing the
```

arrays are determined by the constructor used to create the strstreambuf (see the description of the constructors later in this section).

static mode
means the strstreambuf has a fixed size that does not change. If the array associated with a static mode strstreambuf is filled, further writes to the strstreambuf may corrupt memory. Be cautious when inserting into static mode strstreambufs.

NOTE: Do not confuse static mode with the static storage class modifier.

class strstreambuf defines some member functions of its own and also redefines several virtual functions from the base class.

Parent Class

class strstreambuf inherits characteristics from class streambuf. See the description of this parent class for the details on functions and operators that are inherited.

Constructors

class strstreambuf defines four constructors:

```
strstreambuf::strstreambuf();
```
creates an empty strstreambuf object in dynamic mode.

```
strstreambuf::strstreambuf(int len);
```
creates an empty strstreambuf object in dynamic mode. The initial allocation uses at least len bytes.

```
strstreambuf::strstreambuf(void*(*a)(long), void (*f)(void*));
```
creates an empty strstreambuf object in dynamic mode. a is the allocation function to be used to do the dynamic allocation and takes as its argument a long, which specifies the number of bytes to allocate. If a is NULL, the operator new is used instead of a.

f is the deallocation function, which frees the space allocated by a. f takes as its argument a pointer to an array allocated by a. If f is NULL, the operator delete is used instead of f.

```
strstreambuf::strstreambuf(char *b, int size, char *pstart = 0);
```
constructs a strstreambuf object in static mode; it does not grow dynamically. b specifies where to start the array and size specifies the size of the array, as explained in the following list.

- If size is positive, the array is size bytes long.

- If size is 0, the function assumes b points to the start of a null-terminated string. In this case, the string, not including the \0 character, is

```
         considered to be the strstreambuf.

       · If size is negative, the strstreambuf is assumed to
         be indefinitely long.

    The get pointer receives the value of b and the put
    pointer receives the value of pstart.  If pstart is NULL,
    then storing characters in the strstreambuf is not allowed
    and causes the function to return an error.

Destructors

  class strstreambuf defines one destructor:

      strstreambuf::~strstreambuf();
    closes the strstreambuf object. The destructor causes any
    memory allocated for the strstreambuf to be freed.

Non-Virtual Member Functions

  class strstreambuf defines two non-virtual member functions.

      void strstreambuf::freeze(int n = 1);
    controls the automatic deletion of the array.  If n is
    nonzero, which is the default, the array is not deleted
    automatically.  If n is 0, the array is unfrozen and is
    deleted automatically.  The array is deleted whenever a
    dynamically created strstreambuf needs more space or when
    the destructor is called.  This function only applies to
    dynamically created strstreambufs, it has no effect on
    statically created strstreambufs.

    If you try to store characters in a frozen array, the
    effect is undefined.

      char* strstreambuf::str();
    returns a pointer to the first character in the current
    array and freezes the array.  After str() has been called,
    the effect of storing characters in the array is undefined
    until the strstreambuf is unfrozen by calling freeze(0).

Virtual Member Functions

  class strstreambuf redefines several virtual functions from its
  base class (class streambuf).

      virtual strstreambuf::streambuf* setbuf(char *p, size_t len);
    tells the strstreambuf that the next time an array is
    dynamically allocated it should be at least len bytes
    long.  p is ignored.

      int strstreambuf::sync();
    returns EOF.

      virtual streampos strstreambuf::seekoff(streamoff offset,
                seek_dir place,
                 int mode = ios::in|ios::out);
    moves the get and/or put pointers of the strstreambuf.
    See the description of streambuf::seekoff() for
    explanations of offset, place,  and mode.

    If the strstreambuf is in dynamic mode, this function
    returns streampos(EOF) to indicate an error occurred.

    If either the get or put pointer is moved to a position
    outside the strstreambuf, or if the put pointer is moved
    for a strstreambuf that does not allow output, then
    streampos(EOF) is returned and the pointers are not moved.

    If place is ios::end, it refers to the end of the array.

      virtual streampos strstreambuf::seekpos(streampos pos,
                 int mode = ios::in|ios::out);
    moves the get and/or put pointers of the strstreambuf.

    If the strstreambuf is in dynamic mode, this function
    returns streampos(EOF) to indicate an error occurred.

    pos must be a value returned by a previous call to
    seekoff().

    See the description of streambuf::seekpos() for an
    explanation of mode.  If ios::out is specified for mode
    and output is not allowed for this strstreambuf, then
    streampos(EOF) is returned to indicate an error occurred
    and the put pointer is not moved.

See Also

  class strstream
```

1.365 class iomanip

```
class IOMANIP-Provide manipulators

Synopsis

  #include <iomanip.h>

  #define SMANIP(T)  __smanip_ ## T
  #define SAPP(T)    __sapp_ ## T
  #define IMANIP(T)  __imanip_ ## T
  #define IAPP(T)    __iapp_ ## T
  #define OMANIP(T)  __omanip_ ## T
  #define OAPP(T)    __oapp_ ## T
  #define IOMANIP(T) __iomanip_ ## T
  #define IOAPP(T)   __ioapp_ ## T

  #define IOMANIPdeclare(T)

  IOMANIPdeclare(int);
  IOMANIPdeclare(long);
```

```
class SMANIP(T)
{
public:
    SMANIP(T)(ios&(*f)(ios&, T), T d);
    friend istream& operator >>(istream& i, SMANIP(T)& m);
    friend ostream& operator <<(ostream& o, SMANIP(T)& m);
};

class SAPP(T)
{
public:
    SAPP(T)(ios&(*f)(ios&, T));
    SMANIP(T)operator()(T d);
};

class IMANIP(T)
{
public:
    IMANIP(T)(ios&(*f)(ios&, T), T d);
    friend istream& operator >>(istream& i, IMANIP(T)& m);
};

class IAPP(T)
{
public:
    IAPP(T)(ios&(*f)(ios&, T));
    IMANIP(T)operator()(T d);
};

class OMANIP(T)
{
public:
    OMANIP(T)(ios&(*f)(ios&, T), T d);
    friend ostream& operator <<(ostream& o, OMANIP(T)& m);
};

class OAPP(T)
{
public:
    OAPP(T)(ios&(*f)(ios&, T));
    OMANIP(T)operator()(T d);
};

class IOMANIP(T)
{
public:
    IOMANIP(T)(ios&(*f)(ios&, T), T d);
    friend istream& operator >>(istream& i, IOMANIP(T)&m);
    friend ostream& operator <<(ostream& o, IOMANIP(T)&m);
};

class IOAPP(T)
{
public:
    IOAPP(T)(ios&(*f)(ios&, T));
    IOMANIP(T)operator()(T d);
```

```
  };

  SMANIP(int) setw(int width);
  SMANIP(int) setfill(int fill_char);
  SMANIP(int) setprecision(int precision);
  SMANIP(long) setiosflags(long flags);
  SMANIP(long) resetiosflags(long flags);
```

Description

The IOMANIP.h header file declares some predefined manipulators, as well as support functions and classes that enable you to create your own manipulators. A manipulator is a value that can be used to effect some change to a stream by inserting it into or extracting it from the stream. For example the flush function is a manipulator of ostream objects:

```
    cout << flush; // Causes cout to be flushed.
```

In fact, any function of one the following types is a manipulator:

```
    ostream& (ostream&)
```
is a manipulator for ostream objects.

```
    istream& (istream&)
```
is a manipulator for istream objects.

```
    ios& (ios&)
```
is a manipulator for istream or ostream objects.

You can also create manipulators that have arguments. The IOMANIP.h header file defines two manipulator-creation classes for each type of stream (ios, istream, ostream, and iostream). One class has a name in the form xMANIP(T) and the other class has a name in the form xAPP(T), where T is an identifier that names a type (such as a typedef name for a class name) and x is a letter such as S.

For ios objects, these two classes are named SMANIP(T) and SAPP(T).

Predefined Manipulators

The predefined manipulators defined by IOMANIP.h allow you to control various pieces of the format state of a stream. These manipulators are described in the following list.

```
    SMANIP(int) setw(int w)
```
returns a manipulator (an SMANIP(int)) that can be used to set the width() value of an ios object.

```
    SMANIP(int) setfill(int f)
```
returns a manipulator that can be used to set the fill() value of an ios object.

```
    SMANIP(int) setprecision(int p)
```
returns a manipulator that can be used to set the

```
    precision() value of an ios object.

      SMANIP(long) setiosflags(long flags)
    returns a manipulator that can be used to set the flags()
    value of an ios object.

      SMANIP(long) resetiosflags(long flags)
    returns a manipulator that can be used to reset the
    flags() value of an ios object.

Examples Using Predefined Manipulators

  The following example transmits ********27,00048:

      cout << setw(10) << setfill('*') << 27 << ',' << setw(5)
     << setfill('0') << 48;

  The following example transmits 32,5:

      cout << setprecision(2) << 32.1 << ','
     << setprecision(0) << 5.3;

  The following example sets the skipws bit in cout's format state:

      cout << setiosflags(ios::skipws)

  The following example clears the skipws bit in cout's format
  state:

      cout << resetiosflags(ios::skipws)

User-Defined Manipulators

  As well as the predefined manipulators described in the previous
  section, the IOMANIP.h header file also provides the means for you
  to create your own manipulators.  It does this by defining a
  macro, IOMANIPdeclare(T) that when invoked with a typedef name for
  T declares the following classes:

      SMANIP(T) and SAPP(T)
    are for use with ios objects.

      IMANIP(T) and IAPP(T)
    are for use with istream objects.

      OMANIP(T) and OAPP(T)
    are for use with ostream objects.

      IOMANIP(T) and IOAPP(T)
    are for use with iostream objects.

  class SMANIP(T) and SAPP(T) are explained in detail in this
  section; the other classes are very similar to SAPP(T) and only
  the differences between them and class SMANIP and SAPP(T) are
  noted.

  If you are going to create new manipulators using the various
```

```
xMANIP(T) and xAPP(T) classes, the classes must first be defined
for a particular type name.  This is done by putting the following
definition in any module that uses the xMANIP(T) or xAPP(T)
classes for a particular type name:

    IOMANIPdeclare (type-name);

where type-name can be any valid type identifier.  Because int and
long are the most commonly used type names in manipulators, the
IOMANIPdeclares for these type names are included in the IOMANIP.h
header file and your program should not declare them again.  If
you need to create manipulators using the xMANIP(T) and xAPP(T)
classes for type names other than int or long, you must include a
use of IOMANIPdeclare() in your module.

For example, before using xMANIP(T) and xAPP(T) classes to create
manipulators that accept char arguments, the xMANIP(T) and xAPP(T)
classes for the type name char must be declared as follows:

    #include <IOMANIP.h>
    IOMANIPdeclare(char);

If you need to create manipulators that accept arguments of more
complicated types, like char* arguments, you must first declare a
typedef for the type, because IOMANIPdeclare requires a
single-word type name. For example:

    #include <IOMANIP.h>
    typedef char* STRING;
    IOMANIPdeclare(STRING);

class SMANIP(T) provides a constructor and two operators, as
detailed next.

    SMANIP(T)(ios&(*f)(ios&, T), T d)
  constructs an SMANIP(T) and returns a single argument
  manipulator by collecting the function f and argument d
  into a single manipulator value.  It is assumed that f is
  a function that changes ios in some way using the value of
  d.

    friend istream& operator >> (istream& i, SMANIP(T)& m)
    friend ostream& operator << (ostream& o, SMANIP(T)& m)
  enable SMANIP(T) objects to be inserted into istream
  objects and extracted from ostream objects, respectively.
  They each use the values f and d from the SMANIP(T) value.
  They then call f(myios,d) where myios is the ios part of i
  or o, respectively.  It is assumed that f is a function
  that changes ios in some way using the value of d.

It is often easier to create manipulators using the applicator
classes, in this case SAPP(T), than to use the xMANIP(T)
classes.

class SAPP(T) provides a constructor and an operator, as detailed
next.  SAPP(T) objects make it easier to use SMANIP(T) objects.
```

```
  The Examples section gives an example of using an SAPP(T) object.
  The members of class SAPP(T) are:

      SAPP(T) (ios&(*f)(ios&, T))
    initializes an SAPP(T) object to contain f.

      SMANIP(T) operator() (T d)
    creates and returns an SMANIP(T) object using the f from
    the SAPP(T) and the d argument.

Other manipulator classes

  The rest of the classes defined by IOMANIPdeclare(T) are similar
  to class SAPP(T), with the following differences:

      for IMANIP(T) and IAPP(T), f has type istream&(*f)(istream&, T)

      for OMANIP(T) and OAPP(T), f has type ostream&(*f)(ostream&, T)

      for IOMANIP(T) and IOAPP(T), f has type iostream&(*f)(iostream&, T)

      IMANIP(T) does not contain operator <<

      OMANIP(T) does not contain operator >>

Examples of User-Defined Manipulators

  The following code creates a manipulator setwidth, which works
  like the library's setw.

      ios& setw_func(ios& i, int w)
      {
    i.width(w);
    return i;
      }

      SAPP(int) setwidth(setw_func);
```

1.366 c operator precedence

```
Operator Precedence and Associativity

   All operators in a block (between two lines of dashes) are of equal
   precedence.  The blocks are arranged in descending order (in other
   words, '!' will be evaluated before '*').

   Associativity is the order in which the parameters of that operator
   are evaluated.

   Operator Description       Associativity
   -----------------------------------------------------------------
   ()          function call                         left-to-right
   []   array index
   .    structure or union member
   ->   pointer to structure or union member
```

```
    ------------------------------------------------------------------
    !    logical NOT       right-to-left
    ~    one's complement
    -    unary negation
    ++   increment
    --   decrement
    &    address of
    *    indirection
    (type)       type cast
    sizeof size in bytes
    ------------------------------------------------------------------
    *    multiply        left-to-right
    /    divide
    %    modulus
    ------------------------------------------------------------------
    +    add         left-to-right
    -    subtract
    ------------------------------------------------------------------
    <<   left shift        left-to-right
    >>   right shift
    ------------------------------------------------------------------
    <    less than       left-to-right
    <=   less than or equal
    >    greater than
    >=   greater than or equal
    ------------------------------------------------------------------
    ==   equal         left-to-right
    !=   not equal
    ------------------------------------------------------------------
    &    bitwise AND       left-to-right
    ------------------------------------------------------------------
    ^    bitwise XOR       left-to-right
    ------------------------------------------------------------------
    |    bitwise OR        left-to-right
    ------------------------------------------------------------------
    &&   logical AND       left-to-right
    ------------------------------------------------------------------
    ||   logical OR        left-to-right
    ------------------------------------------------------------------
    ?:   conditional       right-to-left
    ------------------------------------------------------------------
    =    assignment operators      right-to-left
    *=  /=  %=  +=
    -=  <<=  >>=
    &=  ^=  |=
    ------------------------------------------------------------------
    ,    comma         left-to-right
    ------------------------------------------------------------------
```

1.367 formatted input specifiers

```
%' specifiers for formatted input, according to the  ANSI  standard.

   Specifier     Meaning
   ---------     -------
   d,i       signed integer
   o       octal unsigned integer
   u       unsigned integer
   x       hexadecimal unsigned integer
   e,f,g     floating point
   s       any string of non-whitespace characters
   [       any non-empty sequence of characters from specified set;
       can optionally include leading ^ to specify NOT in set
   c       any character
   p       pointer value
   n       number of characters read from input
   %       exact match for percent (%) character

   NOTE:   Input whitespace characters are skipped, unless the specification
     contains a [, c, or n specifier.

   See the Description of  fscanf  for a more detailed discussion.
```

See Also

fscanf() , scanf() , sscanf()

1.368 formatted output specifiers

'%' specifiers for formatted output, according to the ANSI standard.

```
   Specifier     Meaning
   ---------     -------
   d,i       'int' argument is converted to a signed decimal string
   o       'unsigned int' argument is converted to octal string
   u       'unsigned int' argument is converted to decimal string
   x       'unsigned int' argument is converted to lowercase
       hexadecimal string
   X       'unsigned int' argument is converted to uppercase
       hexadecimal string
   f       'double' argument is converted to a decimal string
   e,E,g,G     'double' argument is converted to a decimal string using
       scientific notation
   c       'int' argument is converted to an unsigned char
   s       argument is a pointer to a NULL terminated array of
       character type (C-style string)
   p       argument is printed as a pointer, using lowercase
       hexadecimal characters
   P       argument is printed as a pointer, using uppercase
       hexadecimal characters
   n       argument is a pointer to an integer into which will be
       written the number of characters output so far
   %       a percent sign (%) will be written


   See the Description of  fprintf()  for a more detailed discussion.
```

See Also

fprintf() , printf() , sprintf() , vfprintf()

```
   vprintf()  ,  vsprintf()
```

1.369 glossary

```
 AmigaDOS
 ANSI
 OLD
 SAS/C
 UNIX
 XENIX
```

1.370 glossary - amigados

```
Functions with this portability level are specific to AmigaDOS.  Similar
functions may or may not exist on other machines or in other compiler
implementations.  Consult the docs for the other machine/compiler for
routines with similar capabilities.
```

1.371 glossary - ansi

```
American National Standards Institute.  Functions with this portability
level comply with the ANSI standard for the C language, document number
X3J11/90-013.
```

1.372 glossary - old

```
Functions with this portability level are obsolete calls, only provided
for compatibility with older programs.  They are not recommended for use
in new code.
```

1.373 glossary - sasc

```
Functions with this portability level are only guaranteed to work with the
SAS/C Development System environment.  They may or may not exist in other
C compiler implementations, and if they do, they may take different
parameters, return different values, etc.
```

1.374 glossary - unix

```
Functions with this portability level are based on common usage under the
UNIX operating system.  They may or may not function the same as any
particular UNIX compiler implementation.  Compare the docs between the
UNIX system the code came from, and our documentation to make sure they
match.
```

1.375 glossary - xenix

Functions with this portability level are based on common usage under a variant of the UNIX operating system called XENIX. They may or may not function the same as any particular XENIX compiler implementation. Compare the docs between the XENIX system the code came from, and our documentation to make sure they match.

1.376 help

You have reached this Help window by either clicking on the Help button or by hitting the Help key within the SAS/C Help utility. Unlike other help topics present in the SAS/C Help utility, the Help help topic opens its own window. You must close this window by clicking on the close gadget or hitting escape before returning to the SAS/C help utility. You cannot hit the Retrace button to return.

To quit the SAS/C Help utility, select Quit from the Project menu or click on the close gadget. You may also hit escape.

Most help screens will display one or more buttons as part of the text. Clicking on these buttons will provide further information on the topic listed on the button. You can also reach these help topics through the main Contents screen or one of its sub-screens.

In addition, double-clicking in the help window will bring up a help screen for the word under the mouse cursor, if such a help screen exists.

While in the SAS/C Help utility, you may retrace your steps through the help screens you have selected by clicking on the Retrace button.

```
The Browse buttons will move you forward and backwards between help
screens.  The help screens are usually\LÏ$^3$$\div$wo^íË?ZºÛ*¶íëZËÌÄîô:=Îö$êª$\ ↩
    div$UvUõ¶vNU$\times$??/?\ensuremath{\pm}Ó??êÇ
ë\ensuremath{\lnot}+kl«9\textdegree{}g9î%$\div$\?_?K&kÜVWV$\times$$\times$Õ5««2CÜ{ ↩
    ÎÍÝö??Z0lî¾ÂºÂ$\yen$UiÒàdaw»]ÛÌYG?½»úÚÊÖí?\textdegree{}¡!öN?~$\div$???G®ª®´$\ ↩
    mathrm{\mu}$¯¯tÉ$^3$R'Ey?ù3J1$\mathrm{\mu}$]u$\yen$c¶lY.4?ÜkÛËË$\div$è($^1$Õm} ↩
    ¦LFd¨1ñpðò¤ ú»+[+:ÖNJdÉq?edcã¾1FÕö5õõmX®"tÓ febâeC4$\times$v6?6??(ÈB\?9? ↩
    o¢GA¦kF4z?$A{ú?bªìilkêÜ.£*2$$\yen$"?nnfS$\times$$^2$ÚÚYX$\times$ºpzráI q?ffeååC ↩
    .¶Ò¾¶Ó2?3(Íùàc½A½ucT¤ÇA?£A~ÿ7??!£+*Ý6:\I?Ø'=öJ
,?é»?'ÆL¢S??ì¼??ËÙÚ?Ñ\ensuremath{\pm}q¡¢ID£"??¸i1«$^3$\textdegree{}sUT¾d  D?Y?'À( ↩
    Ê«;JÆ? ú?40?a=ÄÄEvETÑ¡R!Ì¢A«?þ%æ<?ZikY22T?¨R-$^2$>ÉÁÇ£??V«ÍB4YÔ'Å(Osqr'Ï]U¢ÒU? ↩
    Lù?ä¤Iíû3?"?bÅIPâÇH$\div$
iDÊÄÆ{?;
6ÉràÆ" /}?ANô?-üHò'Ú\ensuremath{\pm}Á0ïûïá¦ÞÚçJêHS§Í!Ås¦?\textdegree{};WÙ$.$\ ↩
    mathrm{\mu}$Óil
?1º?+k['$\div$}Øo¡]qu£k/)ì$^1$È?]c\Ê?oí»y??Ú[ =b9Yq§P1Õ·½>¶$  l-$\mathrm{\mu}$?»b~ ↩
    d½::ë?$^2$\ensuremath{\pm}:}Nöd£+.5$\times$N$äeAC:ËK?;Ý^¾???ÚÝj*çeãeÁDöÕönÂëõ¯? ↩
    $ÚÏQ·>Fnl «ÛÕØ»$\yen$$\times$û»X2WÖi5w?<Ë}?¶õUB¼ûºÝ$\div$ä$^1$´$^1$$^1$¯&?ÌØ?\ ↩
    ensuremath{\pm}ª¯a/?Öíä$\yen$Ui¦ÔÚ1B?èSpê$\mathrm{\mu}$¢9Þ¿c?Zúë}=£r¤Â$N?2 ↩
    Aíÿo$\mathrm{\mu}$}?+®ÓÛU?h2?9«´ceßw¯'&ÆÊæÞ)bÍPÖéCyß½Íc½.$\mathrm{\mu}$©JBLªÿ ↩
    ??ÿ?E&v6õ>JE?ÁÂùY$^1$?x8Ñ)ÖÀó?¢?È1  ù¸ø¸X\ensuremath{\pm}h?ÒÎÉ¡ÆDçÁ$\yen$(??ø\ ↩
    textdegree{}Ò¤$^1$Ý{rÉ*???aäCÆÆÆÄ|?$^2$rÔ ,ÆN?¼É8XØï%Ñ,æM£HÃX»ª‘u)Ï\ensuremath ↩
    {\pm}1òÞI¤bælÍ¢?1rí©ô¦¾ÂÀÆËD
0?~sJË’m???û?.*ead¤&4 ÑZ:\?{ÛÌwó04ê3çReW£ÓY6.VUçw
ä$^2$8ªS£F½Õ$\mathrm{\mu}$ÕÅYÓÞ^v¯1á(ÁydÐ?I
muëfB½ìw0óf?yD%?äE?X]m.jØÌÅîv¯$^2$"?Ùu*2s?Ï5Å¾ËafÔ¾Ïbó?â-?JsÌ´$\times$ë·zÆq®û?ì< ↩
    ¨ËÛI7øOh?U§Þlí©Øí^â=¤ÝyEIÅÉ?A´ÝÜØ1Ûì$\div$ð\ensuremath{\pm}$\yen$$^3$diSòð¡${ ↩
    ½Vûcnî}$\div$_»?~SÊBÁÈkm&ïu\ensuremath{\lnot}¶]ÝíÜ’/‘q2$^2$1a¦}n$^3$s\ ↩
    ensuremath{\pm}Ó8Ü»ÁÅ
d©ïò?M-¾m$^1$ÖWÒß¿ÂÍjó??á»q$\mathrm{\mu}$¸i+’?ÿ?"??w$^1$/¦?îãiºÖLY8Xy?$Ñ. ↩
    ÈOÇQ«s´Ó7F?V.>l A0?¯$^2$àM\ê$\times$]$\mathrm{\mu}$ÕX?-þNS$\times$PRlX? ↩
    _EQfg®·rL$ù_=BtWïsa sZíN¿Q‘ÁHð >Ëy?L7ÏÞ=í,®$\mathrm{\mu}$6õÓ&4?£Æ??7ùL]h´ú['! ↩
    Æ1.?$\times$\textdegree{}ãB???I½Ý®E]T}*?á<Í}?ü8\textdegree{}ãMH¦.l \ensuremath ↩
    {\lnot}ì???Ïó2àB"D©Ò)s{?KK?@É\ensuremath{\pm}??O¡H0’ä??$\times$ÙÚYÕ¯£2?F&l? ↩
    ÓSJtÙô?8&-¢$\mathrm{\mu}$rjRs/ð$^2$ÈEI$^2$&O$\div$vz=5«cQ{ö,òhÍ2Tì+´A‘{c&¼¾¾È? ↩
    ÌÙÒbÆI}ÆºæS\mathrm{\mu}$p\ensuremath{\pm};ø?S<´gÊ
=’?ÚÝ½Qó‘^bAAs
3õàÉ2»U´$\times$[6$\yen$?"úû*HEÅ)03s_Íg$\yen$Úmõ5ëæ¿Å¼Â£?$Íò?z½¾ÇJä??üif/6LÀ$cô{- ↩
    ®$^2$ÍQr¯ðã?Å$^2$ß?´ÞNwªÚl®?RìP?}ÆÏq¯?#xyxqÌ‘À¨ùx¸Xù$^3$«5{]È,’$^1$8Ï=éÏ\ ↩
    ensuremath{\pm}\textdegree{}pòàoØê\ensuremath{\lnot}WÐ?È{6DÇÌÆÃÉ{9Þ[«·pDpDR2 ↩
    ]:?>?+È¦Új5ZjÃ??*?\textdegree{}58$^1$8XYo£$^1$ÓZ4$\yen$.D¨Ó‘¸©¯10r‘??Ò\h$^1$¦? ↩
    ¼ó?ö*-,mm«$\times$Ð?(róö$Ã?WÙW$^3$¢AÛ¼c?KïðñÞB£?5ÍÉE0åú0óN&&?l?(?ÕË·’"@xàI?óp$ ↩
    ^2$áI@
[·=?Ä?1Ë!?88£LLÖAqE)p‘Åàf>?li´?^Ì?tòØ´fÑMÍÀÆ6???ïçILö?5^M?Ù8¯¦¦giH[j¦ÚÇy1£DåÄJd~ ↩
    Õn?9^ø?ð’ÒL¢¤èó?mTÙÓr$\div$?.:¨¸ê*P$(ÞsW^Êø$^1$Y?çÑP~ÒèË,?ÝW:9)p3?F~?KI ò$$½Õ] ↩
    mkRgÅ|ú?õ?,$\mathrm{\mu}$?$6Û?¦É$^1$ÁEh>êA?ÒB\©Þ¶¾\ensuremath{\pm}\ensuremath{\ ↩
    pm}Ô"À?B4C¼9)%~Î+k\3.lhÑ?$\times$?Hsç¤½Ýs¦\ensuremath{\lnot}??<8É*%&Iò?$\ ↩
    times$V5jºt©$^2$bÌ¤]% JTÄ??UV7jÈ+o??âR ?’Ì~½Û~F@æ\ensuremath{\pm}áG~M*%’.z3% ↩
    ¨ÍØ$$^2$g¡649ADª?èN,¦î?·b(J?P_T$¡>\?$^1$jÑÂäÒ"?
PÓ?¨ñRdÍVD©BdXS£DÓ?~¸^zð$$^3$]Flr£Ì¤‘ iòePI?2byIJ6TÊK¢??M
?X$\mathrm{\mu}$hÈÚ3"Ê%?X/:?¡L$^3$8$?däK^ÁyIK?r?5l?Ó¤??ÔN
R
üi
bÙ\textdegree{}+p.£DÖ-??á?ÚG2l 92g(bö$\times$)6?Ø2h?{F¡.L´=vd£??ø©?½£?äÉ?{?W)2?ð ↩
    ~?$\yen$Ì?JSeÎbÍ$^3$ _ß
&Ç2‘RsçND¦?Ú39)P¿@?\ensuremath{\pm}8ÑE?!cÛ6?!?X$^2$xIÉ???1&?ÌNTX\textdegree{}âÎ¢Ik ↩
    !*?:%ÌÙ??<IAÒÊQÒ¡F??Ë?)IäE1%? R?âNb¼ÂÊ¡&T¸òÒ¢$\yen$"  ¢??fqd©2TÙr§¦?éQE2J5{3Ê$7 ↩
    \ensuremath{\lnot}ä&ÌA:?ù&Q$.bÄÒ?Rtä’L??¨9Yf/‘¸ÂÌ!:)I"P?CH.8Ê$+?
-><Xòe~~c?ÌÂ?Ê?Æ
O?=1?é?Áy¡QD\ensuremath{\lnot}&ñ¢HD?yÆ?{???¡"?0Æ~!BX?rã ,º(H?$¤?H&ÂI Í4¢á&H?DÔ?? ↩
    gQejâ9u)ò¢Ã"læ.0?$^1$E?I¡.Lr$\yen$K¤?äiÆ?iÔ©?¼(á?0.Dã?®\½p?P¦?QM  ?¡Ç?5Çzñm $^1 ↩
    $åÑNhLB4iÅ\textdegree{}=?ìE?$\times$E $\times$\ensuremath{\pm}bC@ÃF
ª48åÃ?Ï?Ô.?ä
???,’ð£
1+\ensuremath{\pm}!scÊ$\div$ÏáÊL??!PÆ4?Ð*Võãøsi.fÑñ«®¤?¼Í{?2?\textdegree{}hÈ
¤?Ü‘?A¡&?Ç5«ä?C?\textdegree{}©1«?g?:*??à\ensuremath{\pm},?,?º3ÜÌ$\times$e¨k?ALÁE?$ ↩
    ^1$vÌ ?ë$\times$¯£Ð)s?åR??sWN[?@-ÍÌ:k2)??$^1$tá?æfe<@6HS?,ðËuÈºÌÍ>G?E?@®ÛU»ªfD? ↩
    m  Ò,ÐE¢Ù:«tÔ$^1$Ñ?=Í|???Q?hx$ãXä"??îcÊÑ¢?Á?Â?7u[\Ü©ñÂy?¤ÏL?P?Ìhê®$^1$Ùá?H?aH? ↩
    ÛéP!WãZÐ?Ò?kù?¦¦?a©E%?8?ù¸19ñ>?Ð*#1??LÆÎ«,kXRH?Y‘??Y)$??í\textdegree{}?ð!?¶£B‘8 ↩
    u?Ïpæ$^2$ÁÚâ(Í?íË<"?$"nC75v?íL???)?4!>u?Î*ëì]\textdegree{}*è%LHØ)7C7U$\mathrm{\ ↩
    mu}$õ ?Ði?fË&dô(ÝÝm[&R  ?É*Tvés?)U]k¶ÇBPS?*KcF?
+;’î½î¶Ü¿íÝÝÐ?????I$)?RI$Óí¦i6ÛmÇ?q¶Ü9$É$Nu®õÌ^éÒõK=öuÌìîY2[m¶IzG#G+¸ÛI&Ûm4I$I$( ↩
    ¢I$I$?????îîîï$\div$ÿíÐ I&I$\times$/vw\Îø?ðû;¾¢}ô5;?{£m¶í$\yen$|?F+3¨?<ÉræNU?5ö ↩
    ?TH B<Éé?:ÒÀh1N2ú$^1$2¤Ì?Ñ»?
Ì¢ÍÒ$
\ensuremath{\pm}S®\ensuremath{\pm}$\mathrm{\mu}$´¯á_$\yen$MU(L’:X½È¶5Gç*¢ÊSQZ3G6 ↩
    }?6¶B???%?ZÙrs$^3$&$^2$sÈ¢?«~ ËG2$^2$Õ’?l\ensuremath{\pm}
(A_?,4Û?À&?P!yÙYSW?Xñ¢$\times$?DzmSDåHP?E?Å"ZiIÒ$^3$?NY§ëÌÃøL?$^1$9Yê
ÎàÂ®ýãØ??E$\mathrm{\mu}$'KÎÉÊEVS???ÏGhÂ¼\ensuremath{\lnot}áH?oß½
°$^1$ø¸<Á½æh?J$\yen$Bp¢ÇVÈ?êÍ"?´??¯?1xð?'$^3$VÍ?©]¡)ãÔy?-}wç©/'?,P=p¼?pßî${+(? ↩
    ¶6@ÍQFKü\ensuremath{\lnot}, , GdãçKIq3b´£XÚZFMY¸$^1$r¦O¨\textdegree{}ÆVf?}$\ ↩
    mathrm{\mu}$½?"dêK?9®ª£lë1â;xÍ?¼lÉs©?Qeaüp?h-Ù+$h%>vC?åb<MÅ»3$^2$$^3$D0ÔUJ\ ↩
    ensuremath{\pm}?ÙCEi$^1$$\yen$?#???ÆÐatùªròòåO?!Ri\ensuremath{\pm}lÁ?  ?Ë¨Ñ$\ ↩
    yen$LBÈ®\ensuremath{\lnot}+?+;@]Rv~|Ê?ÄY?´ò5$X§0«>QB
Ë,l\ensuremath{\lnot}b½9j2åM6}
?Õë8õö1á&Ø!hÁõ
bÕ'X$\times$Çôa/Bq}>u?Wdá}|zè/F0§Í:tÉ´9(?#$\times$@ôæ?¢niYÑy?,hp'&Û
  76Pñ{?8'Ä2ªI$\yen$>K(=?£}
?8(´\textdegree{}Y2$\yen$O?¤©ö?!?ô?¨V½1Fì????Ñ?ÀÄÃ<?0Û-6#$\mathrm{\mu}$èæ\ ↩
    ensuremath{\pm}3 ?Ë"Ze?ä?qW!3$\yen$æçÊ¤)òÛ$«???ú
ÌÏ?ù5a  L®*dø¶q_?JNnXÃ??\ç\ensuremath{\lnot}-la¦~Â»/:pñt?óQY¤?Æ¶·\ensuremath{\pm ↩
    }??ÌüÀüU=??n?Lu$\yen$Å{õ¤óè?????âd?Í?  ùÙy9W?ÍÓ: WZ\ÜYAZN^NP: ñ£ËPçõãË{??ùùy#? ↩
    $^3$ÐDêrePY*ë;8KJÌÉÌ?z¸2LÊM¾\textdegree{}¸?¶?ã?ÊË<\ensuremath{\lnot}MU?]$^1$´ ↩
    ì¼ÙcÁæ?MÏ)q_ouq^ÌÉ4?êjé¡?ÎâêÒ?òó33æ(Ú ?Q$^3$äQd¡?$\yen$#04rÃ]ãÝÊªQ  ?Äìáwe$^1$? ↩
    ©.Dì¿\textdegree{}(Ä<n½2?
?ÈQ?v$\div$?Í¯??6U?;TG¼$^2$i¦?Ú*KLÎ,Ú??>"Q,,í\ensuremath{\lnot}?JÏ£???Å]¤Íeü[?? ↩
    BY4V]§Bt4Ñ?ÆÎ+Ã&zÌ0ÙË
ÓK2=a^,?RTÊcH?å*?%Ô=ð£Xò¯1@[¡?aJt§K~2/|âÖ[ñoÚz§<)ü2?¦O@bZé\textdegree{}3V]?#?Ú¸æ? ↩
    ý1]¦Æ?ã&ô[?¸?Ê\ensuremath{\lnot}<>´ê(\textdegree{}ÅðÃæUf?Æã¨?/á@,?M!,¡XeuQ¨<é? ↩
    °$\mathrm{\mu}$?í?ræ?ª1\ensuremath{\pm}|ýêBÀ??Ê?f\0Âç|2?iJNs2ÈÑT¦që$\times$?¢? ↩
    PNãM?)?Òx?A@pPÇé¿ÒzY\5R«(?èV@aÓ?À"r´j,©m5p$\yen$HËà?Yâ)?8jTAÂs???\ensuremath{\ ↩
    pm}ù20|m$\yen$©??Í?3ì)Qf?*å´°{Pzü0 ??Ì?A?]Zc6Czø4~?'À ?dôÂ¦6TxFÅ?F?T£Iq
é¼?î?¼?E???ÕDÒ?¢2Ê$\yen$Ì??$^2$14?? K??Úà8?@¸ÆpàÉA~\ensuremath{\pm}îh? ü?âneK?@? ↩
    uPá+´
K¶9Ð?
 P6Ê?Íød69Ê?V.'ÉG1? ô?q¨?R?]pËB.???¡0$^3$Î"àQB"¨?$^1$ib%By."Õ?É??«p°Â?£$^1$Jé§h'?\ ↩
    ensuremath{\lnot}¢ø«K?,6?AR?@ëæ5ks,¨Þ|gVÈã3$^3$8?ÅLÎ
?VEÛ?¡??,Z{¤?¨??Þ$^2$$\yen$C?  Qd£'~B?ÃÿUr\ensuremath{\pm}Áiä&ÄWõ?Y]?Á?,5ölC?T@$\ ↩
    mathrm{\mu}$:s
??Ê6  u?"????pF'l??Ã+Ô?ë?i@?Cf?8?¡IR)?èÀh$\div$90pj'?J,\textdegree{}?Á?:8DÑ?ÇsD] ↩
    ¡LAÇ?Î?Ò¶d0'$<?'?q¨~f?/¡¢#Û K?h?'Ã~ÁÁúÇ9Ç$^1$?-H9ËÄ\ensuremath{\lnot}Ùkh?Ò??$\ ↩
    yen$P$^1$Ö 0ã@!A?@$^2$?*
9Î?^»ÎªÆ;$IÈmM2EêTÄù?(?#XJæE
Üocc®?È6ÒPal-:©?F\ensuremath{\lnot}Cm?UÛDQyáG~x£b?
ðF?óÑ?>Z:$^2$j½?|?f=zHÐÍ)(6FÐÂB ?À9Û?rú?,?ßbÖ\öA?â´ñR0xLË{1Aå<?Âh@sÇ?t?BlÚKèD??yâ? ↩
    ¨Î7/??C?J

ù$\yen$p?[Q?~(A+'NLEÐñ$^3$
Ì)?ÂmçLT¼?À¡DãCø{$u?^2ðH©æÔ¢1$?$^1$?z,¡hD´ùô3 â¡aM?´g
7Õi"wÅ?H»DQà®??(Ø93Ú#âÁ¤¶P8ï9¤id9Ø/?È$¦?éBy?DÄ~ ¼?ÅU*NKaÄüK¡?¤Á$^1$Äog*\ensuremath ↩
    {\pm}$\yen$WC9B\ensuremath{\lnot}?ãÃ??c?¦@¼#?s¦hÄ*d
;?IÐ??î4h7?u«Î[ù?%1??ª?sfÇ0@A?
È9BEDdÒ?? IÀæÅ?Û5j?¶J~rð]+??¶øm$\times$æ'VeTC=?Ôt$^1$Úb=Ú+mT??!?¶èõ?6Uã*r'çÒ$ðZ$^1 ↩
    $3'h?7?ª?Q~?Òt4ÎL$^3$ß*è&%L?Ýl??
Û¤ÝÊ§Ç?Y\?¡Qª¼Â
ß¦?Irr£f~?6§>|éå*b??©1!:VèÞ&rð=LWmÂë(?Ü8Á}à¿@P©UÅÌz?^Çz?é_àEá:6?\ensuremath{\lnot} ↩
    ¨AÅRàr àÂD$^1$$^1$?J°m??Ì?Xæ?R?áÆ?'W?U,/K?Rr¯?@?E!?/ åÎ$\times$â¨?Åç?wÚ???? ↩
    ÅlaH¦m:TÁlÎWRh$^1$Ô64ÅgÃ'%s\textdegree{}lTT°?Js^$^1$As5e®åAGåJ2aÎTÿ?å\ ↩
    ensuremath{\pm}cJ«x$^1$Fðð*  ?@$^1$79?É?ÂqcÚL q??s??"??8¤@Öz/?(¼H¦ãXuZA½Ñ+ó%Wò ↩
    ??à9??è$^1$?$ö? ä#ÔÅÒCÆ?"!ù.ìÍ¦j
?%8àf/
?e? '0¡<?/L¾+?1qF0¼iS~  ?AÎ©??qoâ<!??$\yen$Z0ËX%äv(??>ð©®?¸ÑM ???ç.?ùmÒ©S@$^1$}L1$ ↩
    ^2$?ÎÁÆ
'î(écÂZMÞw'&<?Ü?3Ã*g?$\times$?~$\yen$$^1$Ü´\ensuremath{\pm}ÅEy}?æÅârrà¶lÔeC>úei!? ↩
    ÆÈC??é¸?ÒÆðF]¸Kã9Q??^¢á??î?We$\yen$QwHÆ$^1$Øb:°I??2?b1KiSÕLØ?EUa°$\yen$ÒªCKòCEý ↩
    ?À¸{$\mathrm{\mu}$s$\yen$í?4ÅTí0??7+EEÊõ?#zW#+?¦?PÅÊ$\yen$àº$^3$¶§(-2
G#%¨$\yen$A$Ü$^3$b? k k "rÕCq»dHè0gUÝNjvt~WLF\textdegree{}pç+àÒ«ÌX?*ès§G&?¡§?K8ZSé ↩
    ???#Å$\times$?èq9åá\ensuremath{\lnot}l«?ô¤x\textdegree{}T5D¨?£['®5ö4? Á ?j«?Ëb¶å ↩
    \ensuremath{\pm}q^~ÍØ©däT9?]'?ë^3v0q¢ýÙ6výîLÒ$\mathrm{\mu}$?´?øÕ?\ensuremath{\ ↩
    lnot}?HÔ??á¢?b13óGsfhî?HVÎª?hx$?ÛýT0¡'ª?}X§s?=?æ«??V?ÌÕ?r$??;D8¼Uý0ê.@pô&¾r¿kh? ↩
    KÛg?ûS\ensuremath{\pm} àsêÂÌV?rÅ?
ë}Bí'ê¨e?6âê??xq?2H'ãF$^1$2·Ü?#?9$\times$?TºI#Ôuª8õo¤?Ë$^2$j'Äê¡ÉtÃB¨r.¡ª}  ↩
    î¾LËUdî´Õ¼ê?zÎ=Ðç?ôÃæüáÔ]$^2$Tßõ\ensuremath{\lnot}Ö?C@:¢®ÃÐ~UV2l½ý\textdegree ↩
    {}ê_¨:´Üs\textdegree{}u"ñ?hÆÁkv^ 2ÌÉe5?Í5}@çÎÛ)e*9?/´ZGT9-?å?IuSXñé¨~L»Ò?,L©Ó ↩
    /?\ensuremath{\pm}?;eJ¡ÎmSÓ.$^1$´}2ðËà8ØÈTê?V·?sèífSÑ?6ÚÈ;z='çU«D9ªØ$\div$¦?mÇ ↩
    ??qÞ©¢]Mç´?o?\textdegree{}Ïª?ç?ÎÐ?2ÔmØë??Iê¡Ä?)ÓGV?&1X;@ºeIÀä¼!$^1$Ýð??´)$^1 ↩
    $såPÇN~se?.9 ÛN?t?M:¡Îù ?k7?ºDVI:HÛ_$^1$Äuª?ç9æ2Fo?§NtÇW.Ø3?w:MáKª?Âë{ÍõCÝ-Àâ ↩
    /,'ä\ÜÏÛ.¦æ'NgàqíR+.s*ç½ÏäH1V©?\ensuremath{\lnot}?ª®4CWC´?§?Ãê~/?©?/X»Ö"ÕÞ7D9Ï' ↩
    Ëî?-Ô:?ÇJ¡ÇV.¸?èb*)ÖHÜ"8tsC*ÁËV?\ensuremath{\lnot}ï=Àã@Ë©ÕWyT'å]ÔÚx?#
xÁÍÕ?<ê©?ËU:h;Åk1$\yen$Þ~sÎø??HÕ\ensuremath{\lnot}¢ç?Y#öÃæÐwè¤hôÃ´?*¼ÝH? ↩
    öÃº§Ol8âó½è¤uZ1Í4ë?YoÒté?9Ì$\mathrm{\mu}$d©?¦?~ôú\ÎN?Gt´!#\ensuremath{\lnot}?m* ↩
    óhÍGè?Oüª§N3
$^2$  ùÃÒ2·1
ZÖÇCVSiv$^3$ªeJ$\mathrm{\mu}$MN»?èx[ªEYglÐñëu?7ü?jÇJ·#UM
$^1$ckä'åØlgn$\yen$Ò?éZ¸ÔØ\ensuremath{\lnot}?o?NYAÚ#X9Û?:ÝüáÝ+;im?êÚ#$^1$jïl8\ ↩
    ÞªJ·MÀêfõ~õeVaÅI$\div$H:¯[?Ú¸pÈü'ä\ÓR®?E«?ïp9?vªë?UÂÎ¶Úÿ~;«ÎuCo·W?Ç?Ý'çrYÿ\ ↩
    textdegree{}q$^3$?Ñ?w?nu??xî:?FÖ?.sôC«dâô¼Z?béh?Yj"í]??h¤?æ?ÐeÔÛ??\ensuremath{\ ↩
    lnot}??´¾s]wS·?>IW2ßh9Ü$\times$?@Ó?«[òçÿØsÄPsz;?èsú¡ÎnÎÑ¡Af?T8Û&è?/~?{¡Ï¿Hpò_$\ ↩
    mathrm{\mu}$úñfH?~Â?ÕMÐã*n??$¨îºîªd:ÀFwUS\ensuremath{\lnot}?U$~o?á?¸?$\times$ø ↩
    ?8©º§X;ª©ÿ?~ê¦fcY;àÈö?mªêªu$^3$ªZ?zAÐ]??¢õ´?N¨v2¨vQéFÏR_!??%þ?ÝHÙ/ óÍ£?NøÐ. ↩
    £WÁÂíUãO$\times$ôçpÔ"t:$^2$IÒÂÛr¯<¶Ü?7?wÇý~?îánÔÚ?«XE?ò?&4+fj$^1 ↩
    $á´çslE«áã0s5Cò"¿vÂ,tçMÍ\ensuremath{\pm}jëZª$^1$®Ó»??Ã¨?mÔ^¼KBÖÆª¦s¤gCÓ®?: ↩
    FüáÕ4CÚXÁÎizNó£Zô&?A-?ê·$^1$[$\yen$·\textdegree{}êW@8¯Ìæ?N+W??Ê;$^3$õm\ ↩
    ensuremath{\pm}??wP}?í
_j<\ensuremath{\lnot}?½]MGG$\times$àËc?;·$^1$Ô«£?ut|iÜÇ??«í'«ó$^1$jè/Õ?«ìe[ ↩
    G0J¼Îçàæo%$\mathrm{\mu}$n~?ó?tØÑ?YnÉõ$^3$5p8ä+lÎÛË-ÕÙùº¡Ë\ensuremath{\pm}WL?zÙ ↩
    [6xCæ+h$\div$~?nû!Ó?«¨ýR/¶?W9ºÌT´4ÙlÞú?ú\textdegree{}r??Îs~âæGÜ"¸?XÎ6sNsVç/pI\ ↩
    ensuremath{\lnot}õ9Ñæ"-'îsÜ?7ÆæöÑ"ÿs,ôÎ{DT4Hº(_HÇè:;Íý@äöºÖ?Ðk½&é?¨tÿÐr:
?.ïÛ:{Nó¦?´?Çÿ?:h??RÕ?ÐÝHÑ¨iS"L9zÐïÔ?_üÃ$\div$?$\mathrm{\mu}$·;$\div$?ª$^3$? ↩
    ruÚÿ2éßù.è0.$\yen$Ôc=$3óu[$^1$ºÍßA¯þpí?{ÿÒïPÓ~õZ?þ¯¸FGþ¤n·:?ãÎíZ:¼ètùú´Pÿ.v= ↩
    ªKõyúR¡íó~è{>ß9ÞÕ?t?oÕÏî!Ý6Õ¢§¶vÐ$^3$éi~\ensuremath{\pm}úþý~¿¿_ß¯ïÿþ8iÀþ?xÖ\ ↩
    textdegree{} i c [Ü[\ØXYÄ_wyßÿ?n»ùÞ7Ëø¾?y$^3$ñmÒñP\ensuremath{\pm}dî?r   ↩
    ö·¯åÚ@àffqÿáö§íð$^1$Ì¨dæØßXfeÃ;?û<ÃÅÇåï=D)çtrÚ\ensuremath{\lnot}ù_K$\div$<O? ↩
    ¾¶ÈÁáï\ensuremath{\pm}¤ÚZX/q$^2$ð;¾ë$\div$6û?oñcèíþ?uok{.L\ensuremath{\lnot} ↩
    Äzí4?ëö]¯}·ýýOÇÛ$\div$·Þ'Ã{wð5x¸ZÛ¿]L_KØó~ÇãÜúßGéy¸?ãïãñ¿§ØxÆþo'¦=åò?ó?k??gÖôð= ↩
    ÍãùsØæýÕN>=Õ=ý+?í?ë~WÍð<M¾ßý}¿Òýï§ü?}¿ð|M¿ÌøÝÒþÏ¨$\div$ñ¾ÇNwðú;-~?Åä¿Ûmo6_?^ó\ ↩
    ensuremath{\pm}wàk¡x?wÒðü?Îû[ÈØê·¼ïÄ¯%øùy´z?9?{ïî>¦ãq¸þ¯$^3$ö?£îî7?ï?ÿîn? ↩
    þqöÕûÿoÆòÿ/æg?ó@Úê¡%qi?é$\div$WÉ¾ö_àú¼7ÎÏÄËÈÇ.ZÙÏ$'JÏW¦_*îó.n½i=å£ûÕàvh7Þ&ËÿÀÛ{ ↩
    Ë?Îïþ?Ðïûâ~$\times$½¿ñ;Ä|O??»Y¤¯´$^3$RUÆw$^3$.';?_âÞ3»úüwÚüÞw$^3$¼ü
ÏáõüÏéå}æÿÓòÿ?ëËõ¸>o©¾ÃÂßàen\ensuremath{\pm}¢«ÀÇ»ïË¡%såÞEæÊ\textdegree{}?9E?@ÎBÉ? ↩
    ñkº<¦$\div$ÊÞÞ$\times$ò¤û<Ø$\mathrm{\mu}$Þ|jù}¯neáêmí'5ÞüÝ·gÚ½ÒÆ$^2$ù?__àíþ{?] ↩
    ¦ßý??k$\yen$¯$\div$_OÄìOùq?s6ÃöþuÏeÛTí;H~.½?c}ª{À©wÌexv
IÛû»* fÏèfqÿñNày;Ü?îdÐò$^1$sxÒ<½ÎVOìâù_wÅþÁÌÈÜòhyûâñ·qaàa·ù}ÂçïVèc·ËÞp§'a* ↩
    rxRéïñOäqNÎÉÎF[JÆ¸Ï^?ýq?e3Kßaâê?|/ÚßÝø>?Çcâ|ß$\yen$·ú~?yÙö{}¿$^3$þ?Úx}çÊ«?äöïåÿ ↩
    ?¯a·ìþ?Æ$\div$~?¶iáv}¿uqAd??ûâ'¯ìÕ$\div$öûk©$\times$/?Â*]§^?¿??¿õï$^3$ó7?Í$\ ↩
    times$\ensuremath{\lnot}ÆF7'ðLæy^g
OÞú¸Ó?òù|/Kó^ãêú?'ñú>¯ÖÜýÉúüÌ^w¿þÍÇåõ·8?{Èþ¯«üûùo¡Ìètjcq*s¸?êÒö~î?þO/#Ëô:ìÚõ$\ ↩
    div$ÿ$\times$«Ñå$$\mathrm{\mu}$rO??\§?W§ö
ë»Ö½û$\times$½ÇdÌKù$\mathrm{\mu}$,®.|?½¿Ú^éýûýÑÿV$\times$$\div$>ÐÛö½§Íø$\div$?? ↩
    o©ë¾û¼=Í$\div$^ëöþOÑù#eûßÇ$\times$ö?K/óßÑíþ?½Óªø¶+x:Ë$\yen$."é{ Ù\textdegree{}$ ↩
    \mathrm{\mu}${ë?Ù:}F$\times$®ZÎû{»¾SÐÉãXZæñ,óq=?kñ©Êßr$^1$¾~óÕô92qý?; ↩
    ìúßíz_gíæý}ÇÿßüÞnWÚúÇØú¾?î'ëy?¯Kïù
```

```
Ç°ò)ÈÜñ|?E2??ün/©*#Ôü{ëÌ?kä$^2$©iï$^1$L~<j?p?¿·»]$\div$¢Æ\ensuremath{\pm}&lõ
5sÀèi$^3$$^3$£['>Æò¿Uªã$\yen$à5ÏÎ®$^3$öÑé[¯a$\times$é{.ËÞÜìüKX°M®ªË¯ÖZë{ Óö{n·Oó ↩
    {êèû?ß=$\times$lõ¾?{Ûlvÿ·ò6½Í$\div$eð¼
þ«m?[}ï|-Æ$\div$Þ?Ýw6ÒÛÂ½$^2$ù;9VÞ?Â?Þm°òß$\mathrm{\mu}$Ø1®$\mathrm{\mu}$\ ↩
    ensuremath{\pm}½©$\times$iãÁ¸}B]uF®ìò\ensuremath{\pm}?Õ{Y?êØâ6¤Tg6d??#?$0¼ýß+ ↩
    ÍÂ$^2$çú¸9Üî?)?g+$\div$ÁW?ÂüßËþþ?Ûò}/3Íßçdt}-ï«ö}3ÒçþL_cþ?Êû?Þï$\times$Þ/¯Äõ? ↩
    Ýáñü?$\div$Ñàþ
Îg/êñ$\div$[éòÎÈäzûýçèýïçÃ£¿Âás$^2$¼#yÀEí??A?Y¸ØKXaËg£ÉÀ??NÕ?£$\times$[j¯?_®°À\ ↩
    ensuremath{\pm}¯:ï^ÎÓIGYáAw5ö$\mathrm{\mu}$;+$^3$¯v$\times$Xï!imfi<>Ìýw¾ÔÝ|?æû\ ↩
    ensuremath{\pm}$\mathrm{\mu}$ÚEñ½îÏÂù>7¾ítÝËÿGi«ïÿmªùÝÿ¿ÛüËwÏïu[oû=ÿÌïo½$\ ↩
    div$ñw$\mathrm{\mu}$ô;ßþß?c¨ÛvÝ·ð~{ßxÞ?Õû½ýöÖÛUòv°ÿá{Í$\times$q$^2$¿Õj¯àí>?¼$\ ↩
    times$]j¯eÀØ½Ùm¯ÔÅ~\ensuremath{\lnot}í¦Mþ??ê,ße/©§ÊFg\ensuremath{\lnot}Âªw~>
Ãé?Ðß=äïÔÏÂßPÇô=??<¯_?äïeÊãï©ÈÃÂÞùÞ*§?ÍÎü¾ù==ß?wéî¿·Ïò$\div$^?ÏÃñÿ?¡õ7\?3ÍÝ}¯$^3$$ ↩
    \div$·?G?Õès>ç¡ÁÝy¿[ÿß¿èýMÖgâó\textdegree{}7[ìÏîþåx?;ñy¼N$\div$Ôâpý?æñy?ïGèú¸| ↩
    æ'go}\MÏÀàxü·¼|{\ensuremath{\lnot}<øô$\div$Í'ìr?C?6TU©p\textdegree{}ÎÛ+ ↩
    cÜ¾´ìlÒêâÁ\textdegree{}½½¾Úi.âYvö?í$\yen$þ$^3$c¯Öêbë{}}Ûx~/u\ensuremath{\lnot} ↩
    ÚöZMVÃ$^3$Ûÿ¯¸Ùx½¿ió5v??ÁîÁ\ensuremath{\pm}¼ùý}$\times$Ãîî½ç$^2$îö¾ë»íöZîÿkçÞõÞ ↩
    /»ú=Î¯oã{ÎúÞ$\div$Þø?Îðv??ÝÇï~E~äø;Mì|>$\div$Á$\div$Ýko
ói\ensuremath{\lnot}O_Ýií5;;Ë~Ã]?Sïõ7ZÝ¢Ïûh$^1$z¢$^2$úîÅUûCñím[yu í?Ú$\mathrm{\ ↩
    mu}$o!?ò©6Zô¦âäçcòø«?G?7'?ÕÑòp£MÅÞð9Òò?/??ÈóWå}W°Üï9_Õåty\?ßéðñù¾V$\ ↩
    div$£ËÞqq¼äßð$\div$¯Þôð8Û®?W?ý?vW¯ö\ensuremath{\pm}ñø$^2$óx|_[ÌäùÞg?ëq}? ↩
    CêLÃþÞDßcÁáoñ$^3$$^2$w{¿ÊêIàxì«ìñ0·~vë?Âçsqy¼ü?léÙ?íîdÝæ=?Üî?+
?Ý?)£??F?-âóó?®ÏÂçNÿQ*¾Ï?"f6sl3?Î?*?l#?4T´®®·Òk6G¯'ù?{¨ðûëU_©ëöÚ«-Om$^2$Ö_Ýö??»c$\ ↩
    times$ì°û%$\div$°ØÂìþ?Û$^2$¾$\times$w??_s$^2$ìo$\mathrm{\mu}$?Û?ëKq$^3$ï5~? ↩
    ÈÖêîü?$\mathrm{\mu}$»Ý6Ói\ensuremath{\lnot}ð¶mv?
îû¶Ûkû?$e«ì5?$\times$Q\textdegree{}$^2$\textdegree{}$^2$¼$^2$Óêu«í^Ï]»Øjïõ·? ↩
    WZû»ËÛ;?$\yen$Ü$£Ü\ensuremath{\lnot}ÕöÚãaáZÌÕ]1?ï®ÖÄeíÕ½$\yen$$\yen$}ÊP-!W]¼Ã¦¼ ↩
    {?ñjèæ"Í(64Ø$^1$C>Æ;
>®jéQ\½DÑ2v^.>e&21ÞJÌ~??%wÒx|&.V';ëòøøÙsçÎÁõ7?Éôy¸¸¼O;ðò2$^1$ÜüïW{&fç?o/ ↩
    ÌÊâî¸Ò8ÛÜÞv??3£ÌÞräñ$\div$$^1$YgÀæôpxYS9¸?x?9^Ç§ÀÍßâNãrèËõ·ù|ß;?/C?+?7¡ÏÂÁÞôr$\ ↩
    div$Ùt\ensuremath{\pm}$^1$ao?£Üþ~??w>E rñú8ÙçÎèáKÍÍÍæÈ? s$\times$Î=?økRa$^^«wv$ ↩
    \mathrm{\mu}$"iR$\mathrm{\mu}$ÖB{?yvVQõzíeÅÃLÃ¾ì5¯4½\ensuremath{\lnot}k¾âú}ê] ↩
    ØévÚ%þÂê¾æÿmÜl6-ôMeýõ½ïo$^3$ìúîÏ¸¾í<Kø;=®·Yî{½F»Y´í{îÛ$^1$ð;?\ensuremath{\pm}Ô ↩
    [[ì;K}&¶û_$^3$ÓÞë´ÚþÎþÚ$\div$i´¿¸Ò^ëïnûËTîû?ëîbXwwÑ5?m-ïeq? #v?V]uÚ-ÙfæÛKcg¶$\ ↩
    mathrm{\mu}$jÖù¸/luGÒÞm|zMÇ¯ç$\times$¿i?êðm5?ÃVlæÖZtÅâ3Ïä?%«'ég«;+?DêY¤= ↩
    Ax¸ù¼ÎFV??C??FW#Åãdàó$^1$ØÍß#??ÑÃâî¸8\>>?+Éèóy\??½Ýî¸¼¼J9øy?0½GËàñ:?8XXyY}?? ↩
    ûèð00$^2$%SZ?f/#êq\textdegree{}¸?\©9?9X|<Ã§dddæeds\textdegree{}f)+ÎèÊN~tÖçä\ ↩
    ensuremath{\lnot}õYëÑÏÏ\ensuremath{\lnot}É´§ÉÉ¢\textdegree{}ÄÆ¼ú*=[:Nn\à?¼hR%0$ ↩
    \yen$%ß%·ï'öÓi´Ï¨DÔTôWNÍê\ensuremath{\pm}ôÈ
?Öñ?nå?oág?©xÝÎå?-c k»ö$\mathrm{\mu}$ð?Ç{KK®¦»¦ì-BÒ$K½f£Oug?ä?ïë$\mathrm{\mu}$·ñ- ↩
    t$\div$V?Fô$\times$\ensuremath{\pm}>®?æçKou©¼\ensuremath{\pm}\textdegree{}Ôv? ↩
    »ýMí$\yen$î»U$\yen$Ôv:Û?ÖQãYßëu:«ëîÓ\ensuremath{\pm}½ÓÜ?Òilo$\mathrm{\mu} ↩
    $Úm¯ÒZêzû{Kýn?þ$^2$-ö?ëKkÍn¾æÏ®»ym$^2$¶¶·mPëàBý%ª@c{y??èâîâ_cEXw?ÕÏl£2ôü°?0?? ↩
    V4Ü$\mathrm{\mu}$?EI?eU)1N?ò)Ó æ¦P<2Ö'Nn~V?X$\mathrm{\mu}$¦<$\yen$çÈÂÆeæF%J? ↩
    ùdÊIY3qx$\mathrm{\mu}$39?éH¾Nû?nG/ÐÅÆËÂÊ*^.?ÿÕÀÃåd©7-uy¼.?ÐÆÌÄãáeàfäð¸????x¸Xün ↩
    ?C+?TÊædbs°9?8xø$^1$?ù9|Þ7#?Ðç'ça'dÐè'gHÈçåfsó"?ùì\Îf#+RM?h4~G*C?Üú?ç$\mathrm{\ ↩
    mu}$?æ'A>|£Þ@\textdegree{}??Yô?"$\div$ì¿\;¨Ø[FÂ1'$\mathrm{\mu}$?°? «$\times$Õö ↩
    ?-í"Øi¯ã¾¶ÓÜØWÞÙYYÇ?I$\yen$öÒÍn®bZk.«¡YYÀ$^1$Ô$\times$$\times$j$\mathrm{\mu}$Ö7 ↩
    ?éXiZ?{{Ëý.ö$\div$Iog?¾½ý¦Çc¯¿ÔY õõ$\yen$íæ$^3$gtúÎÒÛYy?èö=¶¼Òö?--u¦3ø6?$\ ↩
    times$zÍ]Ý8ÐÑa?ÊòÒ(
ø1Þ<®a?¾Ö??@eäxãY&ø?£k?x$\times$? Fj
<hÖ6~ñ:èO{1SF<f?J ë?}D?§?$(?¨ª?¨À=ÿfK+9W¯b?4Û¨$^3$KÑR]??QE8
°¢ÄÕZ^m:RçÑ2\éY8ùÙ$^3$¦ãæÐ.??jv>OG9äcbæ'sqs1óÕÉÄC?Ëáâäsr$\yen$TnØêbaI/3&v?Ùò2¤3 ↩
    ;//?#2d<?50ð©çäãKÍãå??Ùê)=Z|<¼,,ê??? yÙyY9$^1$$^2$a\ensuremath{\lnot}ÐÍÎ^QÊäÈ$\ ↩
    yen$'?ViÒRÉS¦L¢)°-P]z4ç\ensuremath{\lnot}s3ª?.L?\textdegree{}$¯?¤$\yen$¡
fÀ0$\div$"ò?ôXF5uÕêè,¶$^2$~õ?$\mathrm{\mu}$ö?_'\üë?]%ÌxÖ??9;=Eõö$k?¯"ÜÙÜêlÞÛÚ?_q?? ↩
    ¤w\textdegree{}!Yé$\mathrm{\mu}$?»Ë+8Ño/4öK{?ý8é·_$\yen$½Õ]ÙÆ´ÒZAi_?ï[u??{ö#\ ↩
    ÚWZém'Æ\ensuremath{\pm}ci?4K?Øðíà#i??ëhñ B®\ensuremath{\pm}$\div$&#?"p R¤Ô  ↩
    ¾a8ÏÓ?jÓ?L{Ï#A4ÛWA¡4û??KÖÈÅ<\ensuremath{\lnot}æ?ðEX3(?s;kSa?3ó?be5~ýú´Ïbf|Õ?añ% ↩
    óæ?sÍ:åJÔR|©Y?~¦F'ååfãcKB?&,ùS¨P&U9´¦âdÐ'3#?6l\ensuremath{\lnot} ?eæeåN¡" ↩
    üÉrsdætr'Ì%Zr2ñðle)6ô$^3$ódL*?*\p¢?>T©9rURVDñÛ+>\ùôéÐ¢©D?gJ$\yen$Id1Q$\ ↩
    yen$çK¤Âí¨ª½Hô?ôRuIóXx??>¤êr~tÆãñ[*õg$^2$u?'A}?*?g?(?~záÓxýéï¢{oNFÎÎ,xè*Ëh@xþ=$ ↩
    \yen$t(0¡Ã|\ensuremath{\lnot}?È¾®þº=}tJúèñá$\times$<{]??XÖvV¶¶/?¯\ensuremath{\ ↩
    pm}M+?k;«Û{K{Q5Ð~ÙÚYZé-l®m\ensuremath{\lnot}.\ensuremath{\lnot}àÅ?-$\mathrm{\mu ↩
    }$¼SB\ensuremath{\pm}?ÖÈcWg$\yen$$^3$¤¿$^2$xÏbZYÄi??8  D?Î:?£YÄ|ú$^1$úO?|õì$ef  ↩
    ¼yc??+Þ?O?Mè$\div$ëÚç%?$^1$
4Ç¿päPm$\mathrm{\mu}$)ùâ,«AfeÈ]5Võa'QtÕ$>]6(eÈBtÚ3äO$\yen$H½{Rr$Ì.vho5*+2d$^1$aåM* ↩
    R$\mathrm{\mu}$$fJ?Ú-4rôiR$\yen$C3:U? $\yen$)ãêT£"IrgÓ$\yen$LMD$MëtæÈNw¤èÎ> ↩
    ÉógÔ8è'L$\yen$??!¦¨=/B??Z +EVèÉ¨Õ??=jiþ$^1$ój\ensuremath{\lnot}Äùáú½Mz?B¼a?Aw  ↩
    ,K$^2$z-¯=a§é4-´L%??ÃËÓÚë~?~r?a?/_ÁýuZIúh¦o?Y'Ðßªªo R}??Ø/Þ??ì7½ñ´d$\div$ñSN?$? ↩
    I4ÎlõÓx{È\ensuremath{\pm}^<x{$\times$È\ensuremath{\lnot}?oF!g Û <jg*sÇÇr©?$^1 ↩
    $Ï9ø?ÛTø'(¸?QFDic~"$^1$É´1L\èqÀ©Óç+Ia?z??ÓªtTYt^=ú\ensuremath{\lnot}é3Î??p?n?Ú ↩
    ?3?æNV5?:?&<8y6$\times$M$Ú:?
èÇK¢?phH?zëÓ£FzÏDq(=ig>|ªè®£?%
Y«*sñ oË%ìú=zLNËªi?ãa´Eêô[  Ne$\mathrm{\mu}$Ø84F@t\ensuremath{\lnot}¡ÍÑiûózq| ??/ ↩
    ýß?ÚÃH.ÒO?CuE?x£g??z«ù6'(?ðÂ">'àÕX9j-6$\mathrm{\mu}$C¶ö?TCEaJm¼?ÑÐ??{3X~tõ^5= ↩
    cÄw K»KrH?.(=8SD6´?êðæ?iV9F^Á9VB"¸gÇ<E£<åU<?Ç4ºÍ??¿$^3$ÎbçÜá?t©?Ð*??qÏÂ?MfÖ£P ↩
    {?!$\times$?Oá¡cG$^2$DWò?YcÒD?Ö?,&ÄrL$^2$=I-@MÂ?4#¨Q?º?Ð7-Dçé0ÙÍá|ð&kÚ« põê6o$ ↩
    ^3$Là?È?éÇ2ì.ÒuÐÞ?5%?òr{0?¸;?9¶J'ÈU$^3$Q¤Ä??i $^2$¨$Nq#LªÕ$\times$?PbÌ7? Í¤O$\ ↩
    mathrm{\mu}$?Ý?R¡<B
a:x??¦?j§.¡È?öÄ&f´ £?T?e3N???¶?êíAx3ImÐ8ôÎ$p:?Ê\ensuremath{\lnot}6ñ?Äß@Ha3é?Ùèþ$^2 ↩
    $»b?X:?A$^1$Ú'Tdô'»??ÑQpÜ}  à^'ÓR{??ÀøöêP?Ü r »L\ensuremath{\lnot}Ël,Èb?B|ü~Ñ¶ ↩
    *0;ÐaPA???d¤'æÌñàÉÖD2?¢6TU??BxKQ(oQI6*?¸¼ä'B?Ü|ª$\mathrm{\mu}$?|ôÑ4çÈ/P?I3Z~Ñp ↩
    ??fi^~y?£ä
» Ê??? Â&?b¢~$$\div$?3a??=?/h3I?ÕLä'TçÄÎ?E?H/¡ñ"@Ï?MY(ö?m¢P?[?Ø$^2$ã?¿"?ØA&f!?\ ↩
    oâªç>B\ensuremath{\lnot}Ç2¢¦ìxøBhe´Ô=4ÞZâ¢,$^2$!¶Óx')P½êbÓÏJCIî
Ç?''E¶$\times$U!G?ýÓÏ(?aRm? |
eøÃmãE?oX
EéÊ ?(Ïm"?d)Å@S?DðÕ?d?A?5"j{C?Ó%ì$^3$xxâ4??m?Z" ÿZ£P¨'q;wL?iØ??Ùí(?$\yen$
#àÓE G?$Dq¶Í@g.$\times$Ï!q?ôT?'Èâ3?\ensuremath{\lnot}À_]$^2$Ué?xªbO?Ui?'b?cÉñ* ↩
    ÑÎIàÿ \©\ensuremath{\pm}Ó'ðR)~?$^3$,v5Å??õÁi?Ym$\yen$??&í®Ò#ñl¤?Wè©?K?"éC\ ↩
    textdegree{}Ú9Æ2Ujb\ensuremath{\lnot}6$^2$Á;á9bA1(ÝA4?X????ñ?F$^2$¨p¢ ?<ÝäÕ\ ↩
    textdegree{}(<4Z?LH Ä¾ä?8k&Ô?*ÃL?ÙÞ,?P+%½?T8\ensuremath{\lnot}ðh-!&Ì¶2W\xåù\ ↩
    textdegree{}\?6/øÑÃKS#$^1$'b¯I?xcÉïP?bDIh??.iQ?Ä?\textdegree{}RÌVY*?j ?¤/?0Î, ↩
    hI$\Ö??6?$^3$?ô?Ä¼?¡Ba· Ùª´(%Ä»?3?&Ô$^3$?ûÕ?Ás$^2$gjWí*ºêÑ£?è¤?  *\jP$^2$]¨?Ð!? ↩
    BYX¨&ë?Y??/Æf?óHAøºrÊ´?D?p$¦;´3.HÍ:aKHSØòª\ensuremath{\pm}D$^1$é$^2$,Àô[?7Å\ ↩
    ensuremath{\lnot}?&?¤£?]P['¶i}?+ZÅÝ[D\ensuremath{\pm}Y?Fl?CÐDÒNº\ensuremath{\ ↩
    lnot}\óÌÒ;Nl©tq?r.Ð~?$\div$j?ÚÇ?$\yen$§?Hí?u?¶??'9r«Q®k®x??,$\times$
lÌx*Õ"áñ·?V?Rð& â¶©?¢RXI®J$^2$??"Dkq¡'$\yen$Y?¢NÙ$HJC?¿$^2$l#?m#S óÀ?Ïyº$\mathrm{\ ↩
    mu}$ßE?tl®¶Øk?\ensuremath{\lnot}$^3$g<?»1åPJ¢¦$\div$t:Ó?e0p$\times$[)¶(o* ↩
    æÙKdju¶?y<$\div$'$^3$X Î,Ï$B4?¤ÅÈ)$\mathrm{\mu}$bóW48d??l2??KE5H\textdegree{}©ç ↩
    \textdegree{}(@??1Zã©?\ensuremath{\lnot}sß:??VÎvçM¶h?2H?ÌÃ(??%?â0Áª80WmË?Ñ?] ↩
    ÈáOÑY®î?H_ð?È?!XÍ OEII='!?Fg?rìº?Z  ??Ù¡?$\div$?Ï31)®++;EA\ensuremath{\pm}!6
8ËiÎ\ensuremath{\lnot}?Sà~y«c'½-ñ ==?;Ìë229 êºç?è9$\times$?sm»r?JF?;?*FWÀ5E2@p7? ↩
    L¤Õ£¡É?11·J¸cHç!Hí?"7ì{·À%$\div$?PP
g:{ ??A&põv$\div$???Û»$\mathrm{\mu}$Ú??6Óy0?©D,ÌªT=~Ý¶?.à?Etá^?-?Uà?Ù%DÜ¨\ ↩
    ensuremath{\lnot}A?8(ø¶üWxZ?=$^1$Ïäa#45b\textdegree{}Þ«ûÚh~??{å¤'?Ó?!?½lÛ?9$\ ↩
    yen$Ë2)\ensuremath{\lnot}¢´Tr??d£ý?øàªnði*ÊÚ
ÈÇ?*v?®7)C?Z??óJ?JÀª6ÌÞ«½4ñ4  rÅB?Q)?äCxT?É¦Äý@$^2$.Rð¯¢ãÃ?Ü£?®ù$T*~ÜsGvàb:ï)?! ↩
    ÉzÌöØªÊÏbk·äÌ?/b+PAÍ?QÆìÒ\textdegree{}äjÍÞpFZ4ÍP??q â~?Z_ä¿&ô,ªcÛâfd^õ.ttÅKä?)/ ↩
    $\mathrm{\mu}$$\times$àÂ?ZpÞqük?¸gã_¦èÞª??H<Ð~?Y¤??5!äk+
^R E¾0ÇËEÜ$\mathrm{\mu}$ØÇ&ï<\ensuremath{\lnot}$\yen$
Ø?æ®qÁì??dv·? r 'ä?Å:=HV?·«T?9@LÁÍ.SÑSÿ¼DÇÿÜ{Í2*õ-lh5-<???M???Wn!???my_wonderland. ↩
    readme$\times$?D[Ù´£\$\div$À?3p?áx?q?B/?FM¶VF<??ù$\mathrm{\mu}$e-ÒÛM?Ç?î?9?6 ↩
    ¢ö5ú?Í¼G$^3$?i5U®ÑÈO_?$^2$?y5Ò#T?yqÊéhfðfü?âh$\times$«ÜGv§ÒY$\yen$Áÿ9iz+àÛ??ùm) ↩
    ?V
efa~4ýiÜÅ&X-9ÝaÞE<p,?{-wkËuëB?)??g?¨»®·?$ÐV¤Ê?v?6$^2$Ø(?Ã??«?>Z?eq?ÀÇÃ?% ↩
    ÃèõñrN¡8wwä1ÌS? ·?åèdsM@äU·VÒ?¾)?M!?ì°,(Õ$^1$\ensuremath{\lnot}??&$\yen$dpRx? ↩
    yòJUØ?ÌF$/=1»«JäàÇ«w¨Ù?ÔþéñáP§Ë'W®Éª¢Ñóý?????W??????????b??????????V ↩
    ????????????????????????????????????????????????????????????????????????????????
    D???¤???7?mods??????????????????????????????????????????????????????????~Ê?? ↩
    tF?????????????&W???H?????&X,Ùþ$^2$ ↩
    ?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&?&~?&}?&|?&{?& ↩
    z?&y?&x?&w?&v?&u?&t?&s?&r?&q?&p?&o?&n?&m?&l?&k?&j?&i?&h?&g?&f?&e?&d?&c?&b?&a ↩
    ?&`?&_?&^?&]?&\?&[?&Z?&Y?&X?????????????ÙV ↩
    ????????????????????????????????????????????????????????????????????????????????
    D??? ???1?OPS-ATOM.LHA?????????????????????????????????????????????????????? ↩
    Þ?&~ÿÿÿý!"-lh5-Ã???ä???Ë\textdegree{}1"???file_id.dizX$\yen$?£ZzÕ$\mathrm{\mu} ↩
    $$ÑïÀ?Ö>´èHØyTè?$\mathrm{\mu}$Ã´??ðÐ8(,?D/Æÿ_\ceX]?¤t\ensuremath{\pm}Tj$\mathrm ↩
    {\mu}$$^3$?~ÁÃæH?T?¨??b|Xâ\ensuremath{\lnot}??ð,?È$\yen$$\times$¢¼¤#õòñdPF0?  ¦ ↩
    ?uUæ?Ztè¶¢VâÆ<É?¸m¯m6Vu?/Ïõ·Æcû$^2$£?$\mathrm{\mu}$¨åèx?/ËÒýë~$^3$ÿé?5?qMÝÅ?- ↩
    ìû¯
Ní|Y-r1·«$\times$Íïº©óØê7ou?gÆé«/øÕ?#~-lh5-¾??\ensuremath{\lnot}???D®1"??
Mod.Atomklomp#$?¿~ûD¿$\times$wwwwwweÙv??aj??"
~EUU'Ìc61d$\yen$I?FI?&îæ?ú¿VäÉ&f2M2I&BI?äbÈªªª.
ª¢( ?????XX?ÿþûë?~&FnæEùìíéÛ$\mathrm{\mu}$Ù]$^1$$^1$wfæÉÕÕ½77ª»$^3$|}½½]Þ^ÝÚêÚÊï? ↩
    ug§¼]¾?òg|Þ~ðíV$\times$Wx»ü?îð$\div$½[WÓÝí,¸úÍ?$^3$Ïgô?jzîs=¯o~Îö=w?~Ðv½ÅöùÃa¢$ ↩
    ^3$Gv§£ðÿgÖv9ûçgÖß:í?o|î;®$^3$$\mathrm{\mu}$åû'ç4?gÖö?â?ø?ÝsÜhIl6W·Üg:ÞÓ¾hzÌþ ↩
    {?Ëõ?\½/5¿Çgs½Ä7i.õpUéïf$^3$on$\div$?^êÞäý¿ml?èãôÎ??6õ_ÿÁíÓ)¨æÞ¤Î{xÌZnõ}·z' ↩
    ÏÍ¶rxU?{ÿõú?$\mathrm{\mu}$í·½þÞA[Å©?P
?ÉÔ¸®?ÔHÒ%^äæèòôòÀmÝ^?Ýõííòû}|}~~?$^2$?ëß¡aaG?=ýêÊCd¡h*4_?$^2$gm¾Ç½îV&ìÛç¶aïä?OOè$ ↩
    ^2$kI¯ÿÅ\Ü}7ù?¿»¿ÿ>¦;S1ôd/ÿöù .~ÿTuúÞvÿr;ßvç?ò?L?$\div$ï_¿?Á?ïÂ;Ä]þÎmóÖû7ñgñ$\ ↩
    times$~ÿ´â{üýð<$^3$Yf¨ù¸Z?oÐ?z
o2ý ãÊ9i?«$\div$Dmø}Õ]¤?3qôfçæ¦ú¤çéMÇÒ¦6ý1$^1$úsqôæçê
¿Pn~¤Ü}I$^1$ú£oÕ?«7?Vn~àmû$^1$úÃqõçëM¿Zn~ân>ân~¸ÛõÆçëÍÇ$\times$$^1$?~än~æn>æn~ ↩
    èmû¡$^1$û©¸û©$^1$û\ensuremath{\pm}·îÆçîæãîæçï?ß¼?¼¼\textdegree{}6ý$^1$ûÑ¸ûÑ$^1 ↩
    $ûÙ·ïfçìC?CË¿æ}ÈÐ»ÛûáOÿÑy?ÔÿÿÜ¯û?þï?Iîõ7ÿùtÿùÏûÍûçÿü·ÿ$^3$¾ã½óý¿?ó+,rÿªø1ùÕ¼ýÿþ ↩
    ?}7<ÿú¸ÿè¯þþu??@*ÿbÿgÖûè$\div$wüÅÙÿÓçÿx¿ý
öyÿ\ensuremath{\lnot}BEû??íoý·Wÿ¦À?'~ÔåýÉ¿ýkõ^ù?õóÿgî«ß?ÿô3ò/ÿëïñë$\div$ý?ý$^1$ÿî? ↩
    èMÏÎûø7üâ?ä?ÿ#øù?¼¿íÏÿë?ÉÎþ8?r?yëÞmçÜüwRVyû/ÿ$?ñ©Âÿÿð?ÿÇ¿ú?ÿò¯ûsÿùÏÿäý$\ ↩
    times$WÿCøOÿ\ensuremath{\pm}þ?ÿø$\div$ÿC?þgÿ·?ÿæÿÿ2_ÿÚ¡èþÛßë?ÿ^??ðgûüùîy7ì} ↩
    ïsõFãÿ£$\div$¼ßêçÿúÏÐs§ìóqô¦ýÓý/¿3?Z1½ÿ$\div$¯Ì_?Gÿ??_Ôó7%~a3îmýãLïV?Ú¯_?azÈ^ ↩
    ýäï}ã~[úzÿººs$^3$þëò½?ýyÙúïÚßõù$^1$'üÜýaÞûÁª;©;ßPÛ_û$^1$úc$^3$ô§{é>¾ÿ>¯ïWCyû?è? ↩
    $\div$ÏýÌý¶$\div$?ÿ¾ÅþgÞBIðÛ$\div$Âp3Üé\ensuremath{\lnot}ßæ}ä$¿~½güù¸ûù$^1$ü?·ð ↩
    ?çð&ãð&çðFßÁ?ÁÁÂ??n~7?¿?Â
n7?¿Ã??n?n??n6ÿ?n7?Àà¿Á?àÍÇðfçñ?ßÄ?ÄÄá
¿Â?Å?Å?áM¿ÂÞçÿ?K?n?Ìûüy,9·øsoâÍÇù%?müeï?>ÚK?n?ÌûöÒXoñ&ßÆýþgß¶6þ8?~|çÿ~9è~»?ý}?ë! ↩
    ý¶¯ó}ô¿?Ó¨;ßSðoõggî?{ë>?ýÄçÿ®;ß_ðgîgo$\div$C½$\div$_ÿw;?x;ßyþæÿ2~ß¯ï½?½\ ↩
    ensuremath{\pm}&ÿÖ$\div$wù¿ìwÿ?ÛÌßýÝÿÿw?û}ÿï3ÿæìþß?ÿ®ø;ÿÞçëS¿ùå$\div$Yä~ÂÁ? ↩
    ñ¯»5ý¾~ß?oü9ÿ»/ÿö3ù?ÿÿ$\div$ïöùû|ý¾ô$^3$ÿã_óïEçó?·ÏÛçíóýc?þGÿWñª?Ã¿Ûçíóöùý\ ↩
    ensuremath{\pm}ÿ}þüÿöÌú$\mathrm{\mu}$ü?ÿýÿý$^3$ÿþüSÿþÖÿûóÿÿÜÿáýîçÿ?ÿÿíçÿéýú$^1 ↩
    $ÿïKÿ¡ûcÿÞÿïçÿ?¿ÿw?øeÿþ~~bïÞýÿv_ÿ»¿þ?þîðËÿ$\div$sÿ½/ÿîgã?üÅûè:¯é$\times$$\ ↩
    times$ÿ¶'·ö¸ý=?$\div$â¿þ=ÿÐÏöÌ?}½þ3~Ûÿù?ýîÿÿoÈþ#þGñ?ò?ÿûËþÜÿþ$^3$üx¢È1ç¯xð?»û/ ↩
    _ì¿ÿìDÿü?ÿïEü{ÿ¡ÿ*ÿ·?ÿÿû/ÿûÿþ¼Oÿßoÿßÿø$\div$ÿC?ýuÿEÎßò?I$^1$Ùü®vÿ[Îßò?N\íÿ!$íÎ? ↩
    ä$?ß?ßïÇoø#·ü)Ùþ?êoòcñ?óýV?£?ÿý$\yen$ÿU?ú?þÚ?ù{ÿü?ÿÌ>Ûÿ\¸?ã¿ÿ>?ùÿÛ¿ÍÓï?§óq$\ ↩
    div$kÕÿýxÿþöÿr;ûÿÛsÿÿMþ´ãÿÜÿïoõGoõ'?ê?rÿ$\div$OôÇ?þäÿ{¡;ô#¿ø?óç?þäÿ{£;¤8ÿKð/ ↩
    úsÿr?ÿ½¿Õ¿ýÈÿþñþâqÿîGÿ$\div$·ûÛýÐãý$\times$à_$\div${Óüò£c§æGÿõïÞOs¨Ýÿëû$\ ↩
    div$qÿæ=þî?þ·Ù¿cúyþ·ØÏÿðÿÿ|ÿnÿÿû¿ÿô$^3$þn?$\div$ÿúHþÝÿíßþÝÿÿÐÿÿ$\times$ÿïo$\ ↩
    div$¦ÙüÂÞ|ÿýjÞkçÿKë}üsr?òa8?X$\div$ù«Ev=ü?¢ÉÿV>$\div$ã¡/ÿáïÿ^ÿýÊÓK??Cø1öëÿõ\ ↩
    ensuremath{\pm}ÿþ/ÿBCÿ· ?«õG_:ýÈë$\div$nv¿??|ÿ~øuûñÒ à$\div$áN¿À?Ä?ïáN¿;? ↩
    æÞ7_ÿ·ùïýïÿÿQ:{ùQÄ_öÓ?ýfå*¾?$\div$ZoÞ8õ~W$\div$)OÝaäÿõ}û¿¿7?»Óëyxúßcßöþóóù^~¿ú ↩
    {üÿýÝÿÐÙý
ÿ0pÿ [ïçúÓ$^3$ÿºýÝÿöþ_¿Ë7ú?7?ºþ?þ~©[$\div$3óïG?ûü¯Ù?È¿úvëþÝÛ¯ûuÿÿ? ↩
    þÝÛ¯ûuÿþÿÓ4Dbÿzò¿lÿP·û?ýBßì_ÿÛßÿÔ¿ÿ[ÿÑÿçÞ´í$\times$ýºÿ·_öëþÝÛ¯ûuÿ$\div$üÜÿÐ¿å& ↩
    ýÚÿè$^2$wÃSÿ?z¿ú,wl_ã$^1$ù_~f¿o?ý¿öÿÿÛÃÿo?ýºÿ·_öëþÞ?ûuÿn¿ÿý_òG$\div$åÇÿ?Îþ? ↩
    üíþUüwèwÿ*ÿê?ò?H9ÅyWÿPÿæûñÆßýEÿCÎOò¯ý½þ$\times$ü«ÿo¢ç/ùWÿQï=þn~¿ò?HÎßèÎßèÎßèÎßé ↩
    ??Ò?¤ç/ùWç¾®¿KÎWò¯ï[ßý
þ¿ò?IÏÓsþóßßû;ý9Çû:?üã·úãýqÇúãõç?ëÎOõç/úóýÈíþäqþävÿr9ÜÎ_$\div$3·ûÇûÛýÐíþèvÿt;º? ↩
    ºº¿î§/û©Êþìvÿv;»?¿îÇgîç/û$^1$Ëþîqþîrÿ¼?¿Þ?_$\div$$\div$$\div$$\div$ýäåÿy9?Ø?¿Ø ↩
    ?\textdegree{};\textdegree{}9Þ_$\div$£·ûÑÇûÑÛýìíþövÿ{;½\ensuremath{\pm}8ÿ¾?¿ï/ ↩
    ûñÊÿ?vÿ;Â?¿ð§gðÇ/þ?åÿ8ÿNw?õ?oõ?/úõ??êN?ÔêN_õ''ú£·ú£õGoõG/ú$^3$ýYÛýYÇú$^3 ↩
    $·ûÛýÀíþàvÿp8ÿXqþ\textdegree{}åÿXrÿ\ensuremath{\lnot}9_Ö¿Ö¿Ö¿ëNÏÜN_$\div$?ýÇÛ? ↩
    Ïü(©c?ÿLÑiï$\yen$J=¯k3ç´Ø?¿Ù$\div$ËGõoÛçíóöùÿCß»ÏÿWß=ËÏÿZgÿMóöùû|ÿþ9ýùõÉlüÏ?? ↩
    QhëÂÃ½)·ªàn¤ö=ÎJNUsÏGâHê5»©ªÔ.p?ñq*?âd3"BBCAÆC¾b0Ìnv^?}ßhý-¯?Í?$\div$¸(l?KË??9~ ↩
    îû ?n??Å5v$\div$Ñôu99)=ï?zªºÎå?û$\yen$,!0qXXx'M'$\div${Y)-\ensuremath{\lnot}ÎV^ ↩
    r$^2$Îå?t)W=ÀBÂÃ¾<¯JÒ?EO]fÑ?½Õ^S7Èá~Ý[VQNqø?YY|?§$\yen$Õn%çì*==$\times$M Jí3?ì$ ↩
    ?K ¿u«$\times$mw|j[dî¶4*1Äºøÿ?C$^3$Æ!Ùi$}
T´ýsUÈ?¼ácúý?´Íc¡VUê¤t  90ÕS´RÓ5(a¢ãq1Pnéie7?æó})))'«Ùof(k,ÛZ$^2$¦àpeæ§éhçé«í'pX\ ↩
    ensuremath{\pm}J?=~qSF?óü/?¾Ýî$\div$"6Z??I\ensuremath{\lnot}ÔwÝânì4/uØf2q¸HÉ ↩
    ???jªÕKÅ
?¯w eFÏIí6[¡Öõ?nq$^1$ò3ñîór¼?,Ø?s$\mathrm{\mu}$ª #cúï[tgÀÞq&èë©(ÙªcÄc1xÕ4Ûí¦Ãa$^2 ↩
    $Öë6ÉyúÆwH??·\ensuremath{\pm}§ãMÐÖ3j?D?dã?.Hlheå6û=v·[ØB$\times$hþ_yäx/¼´ê´:?$ ↩
    \div$U???øé? e©Ömw\Z?fÆ¯£&ù?!!§Þî$å9?öÕÓvlr??[ç$^2$/$\mathrm{\mu}$2'+Õ$^3$AmSn¾ ↩
    ?7-Õfúcú»?4\ensuremath{\lnot}®þWÉãrl?½CEâñøØØ?T¶´$^3$ó$^3$?SôõöP£BÔu??Ww·Ûî7$ ↩
    ^2$ÓT
S$^1$=?À\ensuremath{\pm}AI$^3$ÃC½«$^1$\ensuremath{\lnot}ßîdö[
}sæ¾ù.Gæw½ã_ñÅ?qqPoxÏë)$^2$Úï7ó?ó[õÙi Ó?~?Ok%ºàÖZ §fí?ü·üÞ2?e?´´ÄýE$\yen$?UÊè?ÜhiX ↩
    ?
ÏÉêu[é~,Õ5¢%Ì'ááðþðl\ÜWQMñ$\yen$$\yen$$\yen$¸ÜêJæVÙ$^2$®ª£äÌËðx???·?n¼ÍÍA? ↩
    þæÒ£ÆÞmää$^1$?û?å$\div$ò:o?å$\div$Ül¸g?ßcÈÅÃ¿Göpüy=¼¤´ÝUÒ?kv8í?gæ{!?W_©Öî%è!gn$ ↩
    ^2$??Õh>H[ülÞB?Õ 4¦ßyÂ£§h$f7?ÂB>;8!$\mathrm{\mu}$\ensuremath{\lnot}¤ãñæ§©+ ↩
    Ñ¸:½___X»®PÖÎ¢{+ºÙiö{IiêJ»?m?Ü$\times$ry.áâ Þ\?2©âðwÛ½Îâs??Äñ$}-$^3$S¾«_ } ↩
    èãaà$^2$?\textdegree{}IxÒ{?"~¸ú%LbrY$^2$$\mathrm{\mu}$wÈí3Y???m+¨§&ff$ê?~pzÃFâ\ ↩
    ensuremath{\pm}x¸Ü4<?Òä¢~sS?ü^4iôüÿ*¢ÆÙªE.????)L  ö·7?Öl2$^3$$^3$$^3$´?$^3$$^3 ↩
    $ec_[UQKGC??AAEC?;9=IZõÃC>:(jÊª3ÆâKKV]VLê½?KY·¨$^1$Xÿ?üße~Ðv}~j??'\ensuremath{\ ↩
    pm}?½meu«TË@ÃÆcrqùÈé£òx¸Ëx??5gaWOKSUSSSWY_eilÒèóT?ÇGZ$^3$$\mathrm{\mu}$eYM?5 ↩
    È¡¦©®$^2$fÑª4î?ÝÞ?^?Ý\+P?[[FV5Õ$\mathrm{\mu}$uUuö, mí ®?£Vù??bè©
¸ Ý=5-RTÕSÛí$\times$?I]et©â?  ê3$^1$í?Aëz\ensuremath{\lnot}¾J6 ?Tè?3ecbÉÁôªÜ? ↩
    ¿A_¡aa Ø\textdegree{}uX$\yen$?Ó6V?$gfþS})¿àÈKÌMÏQSUVÙZÛZ2®©¨§¨¨©ªª«??[adÎâä?%2* ↩
    Õ*T$\yen$BdhOÝ\Z$^2$\textdegree{}®ª¨
?KL?$\mathrm{\mu}$?v ÊÅÐ?ïï+?$\mathrm{\mu}$¶e'?%TM,ª©(fç©ìÐ&Xó?ÄãÁaª´ ,v?¼Ä~;??
üÁÉÁ:&§$\mathrm{\mu}$<"uk??Þ?==0xt^½Á-ÍM?ï?[?~?£»Þïø\YM%==MMEEX Ö$\div$7MZ2£èP¢F" ↩
    T©Ó&J??4HP<ÖæÞÙ$\yen$i0Ú$\mathrm{\mu}$¶?v$\div$HS*uuXè?jjÆVUÕTÔtt?\ensuremath{\ ↩
    lnot}®?8:\ensuremath{\pm}Äcz?ÈÍfúþË?Ïv?no3Êc?ÁÈ.k??zè3!£è?#J: (J~âÎ$^2$vgÂßïw[ ↩
    À1Ëu¼ÞÊËñ¸ó<Då?-Uxg'Ð£JJ$\yen$'ÝS\ensuremath{\pm}?Ó%FPºêå£K{k[[k{¦¨R¨Tà\ ↩
    ensuremath{\pm}ÅbÅjSn\Û?s]\(:Ðú@6? áâ\ensuremath{\pm}8ÌMÔf$^3$c]Nc/?ÆÆá\ ↩
    textdegree{}p¯ì^?*L?[?Ö\ensuremath{\lnot}Ù$^3$$\mathrm{\mu}$$\mathrm{\mu} ↩
    $·irÔùóè?¡º¶?73Æâñ%¸\.??ÂãÌLqÃï/39>,f?\ensuremath{\pm}?j$\mathrm{\mu}$k?.^ ↩
    ½zàÕjª?¡ÕÀ
L\ensuremath{\lnot}\ensuremath{\lnot}+ë«kkì\ensuremath{\lnot}í?@?è??5ººh;
lKèÔ¸º<\ensuremath{\pm}ÃÆã2?>?-ËGÇå2XìV'?}z?L?ö¡ªmíYÙ?¨?å?@G$¨ní,Ao©¨ÁnA|'}h(é)i( ↩
    è¨?áOY'?]ÓQÏ?&?\)À)RÉ@T???\X'edÊÍÀ??"?T?jà$\div$ÂÑ0n$\yen$ÎN¯?L_' ¡/Ð·øhq«ø©Ëã ↩
    ??$\mathrm{\mu}$J@©??ø¢E#KÕi?'FêÜ?Î¾º$^2$\textdegree{}?Ö$^2$¶¾ÆÉ?VÌÃ8¯®\ ↩
    ensuremath{\pm}$^3$¸ ~?£¨/ãÃ$^3$«£åGS ?S¶´\ensuremath{\pm}¯?~¨?T?;k u¨?ö¶¢(\$\ ↩
    mathrm{\mu}$@(ä$«1?5[ô?????????jF/l?]ÝT?ö  þB=?ð,?ö@(ò\ensuremath{\lnot}Q-0
~'YØØØÙ?2?W5<?õmkhÎÕ£T?ÇÆ?ÑÉÍÍÑÑÑÉÍÉÈ?åË??X?Üiqj-S?þ"þÓ?« \textdegree{}eijÌ?ä?hâba ↩
    ??¤=^èìÁëü????%ø*
?ù?§w0E%a"B)ñsqop4?Á$\yen$È.®Z$G¼f???¿¶¸"A?\Öèò???W$^1$98®&?-[\ensuremath{\pm}TÕ
```

```
Dù?NèO)Äü\textdegree{}® ?ë?lÕÁ?rÐ?=fÊÎÑ6©©à?A?$N~¢yÝÝå?~?Ý\>?>C?N?§ÝÓQôA?AÑ#N2tå~ ↩
    «Fø#b [*S$\yen$?E’Pà?Õvxy‘?Ôz‘îîè"õÀºî?~ ?,?,?J-ý©§¦¦¤
?©?®?0??ÙÁ¿?)?Õ?â$^2$¡¿Ü¸???hQR4??ñû?Æ§??1ô?x?:r$\mathrm{\mu}$Ù)0üCØ?i2qNT@£‘??[ ↩
    2?\textdegree{}Ó2ØlmÊÿ?W5 Ü%ùLe???h:5B ãE?e3\textdegree{} 2[D)F4?$dÛDÏÒe´? ↩
    ÂDËáßjAwÀ8
+àCÈS\textdegree{}|P0Ð
y?S@áp ¸[\
?(XÝ?¸?{?U$\AUÈã5Ñ9LC??ß
ÀSgÉ¾n??E?íî?þ(!?À?¨ü#ëâÊaDqJ?  È0$^3$F^·
?rMmt9hpB&º<MJ ?=Pâ.?¢8ø5(á©0S%.Î" â?ÈO?> Áó?¶¢?"¡ZA???ÑBP·\?®#øá2Õä*?ÁÐ¡Ø\ ↩
    ensuremath{\lnot}‘OóÒ%!WùûÜM?M?s?®?HLò+ ì2hrd§â~ËÜI?ÈXj?32d?À$\yen$&‘??j??1X18z ↩
    ?hÃå¸É*q)?úr\à¡"?{[0aexp.??T´t$\mathrm{\mu}$5¶Vv·I?.uy‘Åñü8¤ßáoÐp/??)F?’?jêë ↩
    ??+0g\textdegree{}6|52ÜÝà‘?S¸+LæÚÎÆÂº$^2$\ensuremath{\lnot}y?2h?ã$^2$\ ↩
    ensuremath{\lnot}«®´$^1$j2??oO/L]Þß??ÝÝ?\ensuremath{\lnot}V)?CjÊÄ??+ø.fD*6\ ↩
    ensuremath{\pm}íÉ#ñ0?qe‘
o ª ª ª $^3$ioiicYSWd????zê?p}??\ensuremath{\pm}bÅñõòúøøÁØ?x?RöŸeeUes&SRêæ?öX?ðí?LKK ↩
    *úûPo;*ÚÐnØ4Èør\textdegree{}?â¿>_X½=½?,ø??·uÎ????$\mathrm{\mu}$e_ab??Ã?\ ↩
    textdegree{}el:f¢Ü??R5ge]bÉ‘3f$^1$}=?ääõ-Áô!ÊiÝò
????  ýýùùÛË£$^3$»??/??¡ÅB@?C%u5-8qæ?6n?ìX¿BA>¼$^1$+Iqe[WMA?GQWetZ¤ÈA?P>:$\ ↩
    yen$ÝEøOÀ??Â@\ensuremath{\pm}xutX¤8/ù?¢LûFvvLÚÙ?\textdegree{}áÇel}?CÇÒ¦ki[KCF
v¦¨ª¦º?õ>Ê$^2$¶Í£DH¯sx‘û??Ô<44,?ã»«ñsb$\mathrm{\mu}$*Ô$\yen$H?$^2$4h?[ ↩
    ZWSÑÑSVÙ2ô¡t©?¸q! «©?¤7?(\ensuremath{\lnot}V~sÖebÎé?wØu{c?<?oðð\textdegree{}O\ ↩
    ensuremath{\lnot}?©HPâ.?ªS·´?t\ékR???4H?ÓKZÚ9ùÎDØ@"©«egn+¤¨·JeÚ~{c\textdegree{} ↩
    pW‘ý???????
~?$\mathrm{\mu}$¼9?UI&?J?ýt?Y.$\times$UÔI??v¶?æ~â;okpx?@?ÐýÓ:û6??®Ü\Àø?"PpðÐL@rV\ ↩
    ensuremath{\pm}ÍÙÙÑÙÕxºü5@sD¤~äÎÏÐÒ8ûäçigaacaUJ?i(¨h)¢]}À,óP=A6.¨prwbÿ?  ?? ↩
    ÄÆaba??’?J*J@òp?}J(ÚZWUÔÑ?é·kr<2Z0{;
Ë&Ìë¤?¨Ó+V$^1$ÍÍÙÛÃ?$ãóûü??Øpaqr?ä??!FBt¦m\ensuremath{\lnot}ª)((’èi(é*kkkA?í®5hÊª ↩
    | ?PQRÖ2¸B?;?ðzzbþü?:ì  ?¨?Ð©ÅÔ?Ñ?,????Ø:,N?@B?¨6CÇ®Ôzc]?/0e?4À¸î?Ü\£P¨??dHsÐ?3 ↩
    I?ñ?5?[à????©dw?ÅN??æ¡?@?3Í?; dbç?ëÜU¦kogfÎÔ;HÔÔ$\times$Ú5BÍÙèTN?¸Ñ£?lÍ/CPÄT?Û+ ↩
    øvi?mij.$\mathrm{\mu}$h}?diw$^3$$^3$d’î?û?Ù´@??>¸?Ì :?PîÎÍÈ%ñ?R-ST¡ ~ô?<áÛ,?Øâ ↩
    ?[?’h®«%X?5ÖW?U,ttsr^1ûk2t(\ensuremath{\pm}???hÓ$^3$»ËÈÀ!ótÀ?>ZÂ’ªàçn?@
I?»??’$0êà[à/¤?RnNÕ?lÓ¡?
$^1$it/?ÅàçäiJáî?ÈÒS?ã6-Ú??íÒ?aÑ=$\yen$È ú?åb?èAúMO??àâ???AáÂ=-bPbzÀpÂtêù<x9w?ê2U ↩
    (*Bª«êBîÆâ«Rzä¸~*$^1$W$\mathrm{\mu}$1ªð?0+ó|_;Kô=?£ð¾wáxOèi$\mathrm{\mu}$[? ↩
    £½áLÐ??R0?ÃO[Û´?,$\times$-yNà%ñ4#ÁGú

ãÐg?t$\yen$ü?rèÎ?h¤$}Ê~m?Øí\ensuremath{\lnot}*hgy?ü?öëo%$^2$$\ ↩
    times$ëuÍn¿iôvÿGk®Óù¾’ßø_?YÀªD?z¨GBÕÎnÎÄæ=óAÑLo¡Cÿ½¾ð7©åÝÜXwNPüÂ^^CÄ»¢yÁNeÑ~[$ ↩
    ^1$TK¾]$^3$\ensuremath{\pm}>£Ìy_êåËÝÒÂt3V~Wc¦ô=
.§i½áKñ$\yen$ø-þè;Ýîw[Ýÿ?_;AMYelÕ?$\yen$ï7$\times$ççðêÔ?0¯ðÐØ????]DÄa0QXkº? ↩
    ÃEa0HPxx,???-ê¿_¡! ~XT?t?ú?||(? 0À~qÙÑr¤ù&U$\mathrm{\mu}$Tô?$^3$\yn?®ü#? ↩
    JoåxrÒÜ.?$\div$o$^2$Ôé$|$^1$?.¯o/RÕxuPV\ensuremath{\pm}j$\div$Gph%ôZ???W_ááð?T1 ↩
    +?Ààâ"\textdegree{}øßñ:/Ñ
bº#k?Å?g????Äbq1qx|0?£#1?Di+?5?ÁÃ?0??n|????;û???Úþ?É¾}¿¿ûß??Ó$^2$Ým¶M\ensuremath{\ ↩
    lnot}óÇ??
?e´?tÕ#Áõ©ôªj¼\ensuremath{\pm}|¿?@:ô??ÁË
mXH??®ÿ0m~¾½?\textdegree{}@Õë$\div$V©@<Í
E??®"?+?@E~ª~d¡æ?%mU5??é?,·??þþVW?¼·?W{$^1$ÛlõGÍùò>]·áò\ensuremath{\lnot}-Z-4?ò? ↩
    ïc}~"??õ?,?Ò´sjmBpP´W\textdegree{}J?æ??"$\div$$\times$·R?Ë/¦!Ð|Dó?â?|ãI}1?? ↩
    HÅ©Âõ©í(e¶û
NÒô½-?«Ùí¶ûæóz?¼Ýnw;qOÝàD?áñxüiINNü?Ël??kW»<½ä>?¡â.?.ù?À?Îí]þ?iZÖ?d?_äÒÀà:?ÀC¿Xr) ↩
    \textdegree{}©¦£¡å?ñ/Å
âñeøá?þvtl$\yen$·ûÉ=»W«Öl6»^=?mÊpêÁîúý? Ààð?pIèî?Çãò??~??Æ??4c¢‘?ê20´ÉÄ? ↩
    Q¡ê7d¼ÃáÊEÂ&‘"#øw$\times$çð?¯åÂm_â  4>??ØS$\yen$Õ?2Uü$^1$j¦$\mathrm{\mu}$$^3$$ ↩
    ^2$ÛÍ®ÃW¨Óé´úNÃk·?4ðxdEpFR.ø#0o?AJ?UY^0-È?§VEß??(hR?_Wò¼\textdegree{}æä?# M‘#8& ↩
    è$\div$?$Hä?xÈC}$\yen$Å"©§¤£~åONNNNN?‘#;Èâ
®Ïc\ensuremath{\pm}Ùí·;Þ?"©%ì_?bÀ?`b"|(5)e\textdegree{}Ð?0?AÒæ(Æ? biN???F??Z`?B? ↩
    ÛØq?ä?B2m?ì§"?¿®]Õ$\yen$7UR¢;àa?!>Ð<Ò¶z[o\ensuremath{\pm}õ~æy^>Åñt~?#èzZV·]\ ↩
    textdegree{}Ø1BF~$\times$m¶Û?º¢'è(d~/x/q?è?Z?A"B'hÉ¦?Á?~0{"Aa??e'?Òò©pêÀ<FvÄ?$\ ↩
    mathrm{\mu}$/?¸?/[??qSëC\ensuremath{\lnot}½UUee`EW?Ô ü$^2$?poý?*j_¿Þ?Ëqæ¨*l®R$ ↩
    ^1$ßoÄn~?  #dH¤N ºDÔå¶~OØÒÓ¨~Q¸¸Â'>?(9m7»mCà\ensuremath{\pm}¤^$^2$ìêðAÜôüÙ??]Hë
]twï»&H»,ë9RûÏ£\textdegree{}ÔúzO3ÈÑøàø?áèü¯3Iéz½~Ãg$\mathrm{\mu}$??ÙßÖ>(Ç4?ñ?~åPP ↩
    ?]!"5¯?b~B ?x?É?}!P»ÛPÂâ ÖW\?$ðéBÚ?ðå? N*£~?$ÖdpM?*ro?6?)IS^Íªw60X?\ensuremath ↩
    {\lnot}$LLLV???"q?Á\ensuremath{\lnot}n??]cÉY?C%ÉäòY,G#??¦O%%Þ;?Åâñ_? ?bp8ÂDÅEDà ↩
    \textdegree{}0Á?~@ñ??ó>¾¿??Ò?eð!?JëUO?A»>W? ÊÐaT?5$\mathrm{\mu}$$\mathrm{\mu}$< ↩
    ï?ROc\ensuremath{\lnot}Ôi|ù?/ÈÑø?áø¾?#¤ô´úh?$^2${ä¤¯??¤À?Sscºæfxü~7?[?Ã? ↩
    âqHKÍòg¨(?0A%B\textdegree{}?\ensuremath{\lnot}[xÈg???$\mathrm{\mu}$?\ensuremath ↩
    {\lnot}#?8m?A?O?ª¼äÆC0È=Ã£?Íõô"àÀÑpÏ$^1$!Ø.ºÑª%nÏ\textdegree{}ðø??A??F&;âb7>X¼ ↩
    ^4hÂ?F}r?ú#¡??Äâq?¼>???Ê&"#?Àà/ðp?·Æ,?\èFþr?Áî¤cÓÓÓ?d#Ãx?¼HJCáæ?xFW$ÞåL$\mathrm ↩
    {\mu}$$^2$«£áð7{]¯O¦ô|ù?äù?7äy"Ç>§]\textdegree{}Ùí·?\textdegree{}?øö$^1 ↩
    $ÎHTäÜÈÉáð$\yen$ex?  IMðÐññà??~?ô ?ÐQQQ·pPà?lG.?ÂÔc?$\div$?Î??[[$\mathrm{\mu}$< ↩
    yÅÂÞ?;;Ûè4=$\times$sÌb?-fw;?ëºª??Íæ$^2$Ý?+ÎÛÔÒéu;^%B?¨Åér¸Ø§Õv\Þ¯I##¤Ôí¸$\ ↩
    times$NPqøþ)ÃAº~®Ýë=?3ÈòCþâ^áÂûÈGåcò?Ñ0.É?PËIê´_£ñ¼;O\textdegree{}ÝKÒ3Hë???Êå$ ↩
    ^2$ß?)Ä`~?[ÔÎJíu^ãx~?$|U%¾¤dkÝÿ?É|^1ÌeúL./??ð\ensuremath{\lnot}õ,Ô\ensuremath{\ ↩
    lnot}¿Óó¼?¢ùÚ/?Ãñ¼¿CO®àLPVÛ¤\$\div$  ??Ëæs}fs9ÕuY¤Èâ¢ðPOkÓ?$^2$§äñwÛm~£H?' ↩
    ó¾gßüß>ß¦Õl·?  yÚk?Ö¼?_â£qÝ?_©êúÞÃ=Ïöü$\div$c$\times$g:®£$\yen$ÇôX|??ëºÔ§\ ↩
    ensuremath{\lnot}ë(ç%åw#ÇÎDt?ÍïûîÓ¼ï{Þû¿ùÞ?äy^nO©Øí·|?yÚ*\textdegree{}íð$\ ↩
    yen$sËä&???èú^£7ì3ý¯q¡øÝ$\times$wÝ$\div$}$\times$uÜöý·iÙuùÌßQÊäB]6
þþöìâ
Íu=?ß?½Ûí5àÒ?7ËñüO?Àù½ÿ{Þü¯Íüßò¾WyÞ$\div$ÿ05¨-è}
F«_´Þp%¦'hiëY?u?Kß¡0?1qØÌW/ÔuYÎ»\ensuremath{\pm}ÏöÝÇsÝþWåüIêR?'ä~_åw?CÛö$^1$ñÚù-? ↩
    ÆôX4D<?û?G?îÖÒÖÆ$^2$5Æáð7®Ë]ªÔi´t<Ä?$^3$æwý$\div${Þ|¡ÑAø;Óæ?þ?=ò¤|í&Oêê$\mathrm ↩
    {\mu}$û á¾àËq©hMB?]?ýR$\times$'w§È?H|?T^'?Èt}'KÓæz®¯;$\times$ö9îÓ¶?C»ü½@:º
KÛh3ýa$\times$g:¼Ög§é~,~K?è\ensuremath{\pm}1xX?\ensuremath{\lnot}ÛËªõ¡3P%$^3$$^2 ↩
    $¯«§¤¡  2òÜ?MæèxJë$\mathrm{\mu}$~®£Óú?O<5È+'âx~?¢ùÂy$\div$ýðÐkû¾ïþgÍù¿7Eàx>?l? ↩
    Xo3Íóô]6ÔÔêõû?hFï}Àá?Ð?$^1$ÞU? Dv  7 ?EN?$\mathrm{\mu}$OoÉ?Â{ñXÌvC)?Òe$^2$ý>c3Íõ ↩
    }fw®ëû?Ë=í4?·oÛ$\div$???$^1$øÁ]$\times$t?ïÆî{?·íûn$\times$@?$^1$Ùv=]Öç:¼Þk3éòùn ↩
    +Èãñ¸¾??Áà?"ì?:V:?Aàlñ#HDjzPé?;7È?¼)YMîïs·Ûmv=»YªÔúOM$\yen$ôC|ù$^2$>g?óÇñ´z??Ä ↩
    ??Âð¼??Àm^?W
ûñ<]?ÆñüßÞdçú??úzSÕÕjõÁÄ?:Æ?ëpd~pú?\textdegree{}Ó¼~,??B8Íj26??rÐTîòH?0p¡Çç??? ↩
    ÄÆôX¼f7?Ée:<¯IñrÝ/MÓôýFg©ê@ êúÌæw;ëzî¸?p®Äk$^2$'ûØÛ¯ëÀ?ës$^1$ÜçYÕõ}Vk5Ôæs??ô$^1 ↩
    $oÒGôyLC?Æb£ãØX¨????????ùð???;??bÆz  ó$\times$C9 #·?{Ê?  îLä$\times$"`¦¨p%?¦öà? ↩
    i-¦Ïc\textdegree{}$\times$ë$\mathrm{\mu}$½V§Õõ5?OM¦ú??i=?<??7Íó<Ï3Ëòü~?|üm^P$\ ↩
    times$!æùÞçé4K¦?ïQêzºV¯X?öËy#Iî7; Þ $\div$|-\ensuremath{\pm}âqeÃ¶\Üï'AEIKOQV-®$ ↩
    ^2$$ÝJ#?V*F?åálÑáÓ?ë??Å?sà$\div$E'?¸ÌF$xì¯Èdr -GÇåz@?rÝ/Kéºn§?æ:£¨ÌºÔ?P¶ê?$\ ↩
    div$§éún.?x
?\¨
YLG ??\ensuremath{\pm}\ensuremath{\pm}éÕ?áYÁ$\mathrm{\mu}$?o/$\yen$qÞ+øI$\ ↩
    textdegree{}|ÐM$?Ú1Êêp?hf??33??ø??
[íîóv#dØvÏe\ensuremath{\pm}Ø?â?«S©õ}_SÔÔj4ú}?§éö?é+KwAô??»??O§Ôj=OSÕõu 2?Ë®$\ ↩
    times$ì?4II?Õ·?¯y½ß?ãâÎ·#|à? 830??&Às?¨W$¢2$?Ñ¡.dã?=???Ð?àD|?$\div$?Aú+???H?ìj ↩
    ??èpN?8y?VS)èÀ?<ºé:B¿r½!>ÃuÑôb7?Áí???v'?Àà$\div$?>QQ">`?}(~?<AÁ´'æ ?3#U?6T?©'l? ↩
    KàK\  $\times$R$= ?~"g&æ?/Ä?7?F?$xwÜ?[@I  ' ´?We\ensuremath{\pm}
ØkÂ$\mathrm{\mu}$ÁZÁX~s(mFû\?6À?½Ëd?K¶I-¨u2?$\yen$û Ä!?+Áá?"!ù?ïf?h?@ET  ?\$`Hk? ↩
    È$ôIv©fÐ??&?/ I?I?ì K?.?À(Æ?;óÅÅ½þ&46º!ïyÅH<áK¡?G$\div$?+î7câ??z? 1¾

' P#HG?G:BÉ?É%Î+DRY(
2xgÛ¶%ëÅGêL&HM??p?~IÒ?ÀO~ß?ï?£?p?6Ü;ERCB9í?TTKã;] ?$^1$?7w¼!?(Y"???Rg*<?¨©??Ù.Ø$\ ↩
    mathrm{\mu}$*v dÑÙdá Ò?Æ?àP?P¢!G?($\yen$???N?b??ZWfÑìÿö(®ÔÃàÅ?H?ÁÎ\?`¤A?xIXQjQ. ↩
    úH\LM?MN?N$\mathrm{\mu}$åÅ?âP
  B*x??)ð?`¨?\*??\!@NQP?H?Es$2Á,?·EÖç$\times$¢[>XúdçÀ?S$\textdegree{})PuU?`$^2$fl; ↩
      yrJíæ?ØNMO?·!>Bx% ?ø
```

```
!T?(ÁP¤?D!JWc\textdegree{})ÁÂT?TßÈUV‘0d¢bJ-IX[°?¤1D#?\textdegree{}1¶I?r?$^1$"óAJ ↩
   `&&Ç$\mathrm{\mu}$ëÐÍÕ?4.¢§$’NÈO|Ò"»Â]?B??v:(¢,£)Q@¤!J%1?(Jq*?SSÈU?VB\ ↩
   textdegree{}Jà+Ä\textdegree{}($^3$àJ?B@?ú]
ÿ?|?"?À$\yen$í&(!:ZRRO§ÀP
~
~7?PÛÑ  E?ª0?¨ä)ARK)S?S^¶B¤UHY
À+K+áY??8?l?\ensuremath{\pm}EÇê?F??¨~K  /?T?R?@M?t?ìó\textdegree{}öî?}ÅjÒ???dJ! ↩
   TB0Tb?©?)H©D)¯Jp*?¢Ïb*
\ensuremath{\pm}+? $]¾’?\ensuremath{\pm}E¡EÁ?ÂèË^?\ensuremath{\pm}uÄ!,Q1Æ$s3brwÉ?t ↩
   !??O7HP?~PT2"?wDêB?
BÊR*XËÉ??8$^3$Ô
Ê?D*Ä\ensuremath{\lnot}$|r??,-K¢?F~EÇÕ!.Y3?þa6ùvN<Q>Y@% J
vQ??,$^2$$^2$‘GX¶)nÊag~å?U UB*¨+¯?E\ensuremath{\pm}Å+ôH£(¿?‘¢\É ©$^1$Ã  ÉæÄðôOP6( ↩
   H¨(?gMÍÕÚ¤?R?å?·0?å?*L*1ÝðhÏÃ?Yü©o&QrbÛI.]!éßÊfBh¢núTÌbæÄø$^1$ýÈ¢J??(D¢?bÎ#?Xñë ↩
   #£Ê)?)o
jÕ>D©¡*q´ña
á+Ê,HY?[{???_ÍM??I¢$^2$À’F+p?Ä’ù? ?ÎQDZ£?QÒ???¼$^2$$^2$ì¨0©?¨¢ ´âR?å? Y¾¿6§ò?å~¿? ↩
   jWOIòòs ¢ç·$^2$|x??©?ÈEC-Q
(TaTr $\yen$I,$\yen$åÅ$\div$eAEM $*èKÅ==Ë+ÈX6-=ª3Ã’~$Ðe?·dñ ûÒYÊb¢?:?¢¼(ïJBÊRÊk$^2 ↩
   $~?Cb¤J¡*Ì+?ÌªabØ$\mathrm{\mu}$õªúøÚý´ÿÑÀ^.òBt  ë$^2$|$\mathrm{\mu}$A)PPúÅ??JFÅ ↩
   )jYMzST]%*«b\ensuremath{\lnot}¢¶ð\textdegree{}»¡!gnÈâêx°å?M?M:G><a>Y@YBR¡$^2 ↩
   $1¨$\yen$3¨?,¤æ?¤TÁ¼%1?ìQOvT R%UéVB\ensuremath{\pm}\ensuremath{\pm}‘e?? ↩
   Ä¯Ö7ÆyeÎ2y-N’J?b~?‘Pè!tY
4F?~\ensuremath{\pm}HZ$\yen$RÞÂSÞ?TU?«!XQ[AxYû4?´??ô<$^2$lTìIëÒ|
?
?3?E?áFZ¤?)ä$\yen$J!JFx©?Í{$^2$~JÅQ
$^2$?+[?7õëX?äÖxw,-O??P?cñ$\yen$  ?eF"££ÌT?gy?(ÅKúE70§?~¤Øª(\ensuremath{\lnot}(¢» ↩
   ,ýª¢,0£+?$\yen$»’$^1$?0¡½( J(¯J;ÒÅK)S?RS?ST?©(ª(\ensuremath{\lnot}( 0\ ↩
   textdegree{}úÔga:Q>åvOP^?*(S%ö
?\¤1?)e1e9A

£?\Â¸ÂÇûTgJØ00,~(¡åª ?\$\mathrm{\mu}$F-e3\textdegree{}ÅH1å*UáKxST8¶ÔUq?°$\ ↩
   div$i9u·Â?è¸¡??Âa?dR?b!Q?¢
?ÎTUZÓÀ[m$\mathrm{\mu}$<?mÛo.ûâ]°Ûö;·m$\mathrm{\mu}$¶ïpª¨ª?"1V@P"?È?????ÄH?? ↩
   ¿ýÝ»·w9çÙ$\div$ß?¿]ü?ïO½$\div$k?iÏÑÕäºì[ÿ.ÏöÉ?$/æ£ÿ¼~®þÝ?ût_¡GÿGètöå:RtÙ.) ↩
   ÔeJî§ûº¼$YëWaÿèñîà$^1$ßCÿ?&P½?ötXy.?£Á2é?ïHý]7$\div$tÿ«¨ýNK«ÿ?¯+$?X.ê>Cüýy¸.k7? ↩
   Þ=Ï$>Nÿ·CètP]?Osîü:_@x¦òþ*Sò?ûu?]OèêÿÃ¯ý$^3$õðò}kÏòç¿YAÉt?èÓ¢ý?¢RIJý?4Gÿ?7ý°|§U ↩
   ?¿Û#óùß«¤å9ÿÕÐ?¡Ét_éÑÿñÒ·Kú?ë)¿öéÿË©?¯)ÖÁv_áÚ-þ?åä??ÐçÿÃ¡ÊLDNÑ(ù.Ñ)9.õ??ï) ↩
   dérCÐ$\div$Mþ]?èê=?ª7ôÈïÿsÑæe>C=Ð?èWE(Ù.ýJGìé1j?2$\yen$?Ñ(ð]??GÿûÌ?ó<\ ↩
   ensuremath{\lnot}}LÁÊÏ,?Âl\ensuremath{\lnot}ÃÊÌLóäæ.Vc~úd?Ó2?Þ ÿó!?ù  ÿö ÿÿ?*U ↩
   /9?4éÃ§??Wõ?b$Ê??~$\times$?Îê>Ô??Û¢ÙÃ£UKÏì6?è?½têéé*Tû$\times$?ìúÛ8ºz?sø??!1 ↩
   G¯·¡?asóó $\div$¨?ÝWõ?&H|<òYôyÅÕ?ÃR&Lë¨9Ì+ëÛ6H¢^ka>?G?6jØÔhÕKKW$^3$?¢H¡R§ÝNPr? ↩
   ¸1"©Íáººa}i>tÛ¢I6Á?zý?z
?5/4???JU&H$^3$Ì ã?0ÃQ$MÑê?6q.ªh,eoKÃá4rÒä_sWÈ?Tý«Pp?_??1 óB??gÈF¶FdôY$\times$ºzùø ↩
   ?C?.§é¢çsºzé$^3$>¡ZðH7%0Öú,ß$H8mW’Ænô~ÌÙîé?$Ê?8~m9?Åú?Q?_¾x?wÈ?DÃ?¿ºlÌj$\div$?O ↩
   ~?z^w?ì??ä??]8¯?40??.GNW¨õM9?Aî?//4Ú¼>?~û~!?Õ#þqþgG¨ômI(t?$\yen$Ó§A?I5?>$\ ↩
   mathrm{\mu}$·N?@b?AÂàÔt5tö?©mGun$\times$jH4üüÒb¶Ôê$Dhq?GÄü¸
!á??sðÑ´$L"?_w’!?¦3?ðY$^3$~‘g$^2$û??@D¢^saëÐÎ&D?????õ???ÅCÎ¿b%N§]\ensuremath{\pm}$ ↩
   \times$k§§§*lùãÇ
ªFÄ?ß¾àó$^1$!?äò#?>
’8E7n£ãMHªhAgF??|#\ensuremath{\lnot}Õ
??"k©Ô1TæÃo·45§Ï^¾?\ensuremath{\pm}?BÕ?@£¿??ÔF¡5?áû/~>AÂj?àé\TÑÃZ®U=R¡É*<c?ÀÒ¨|?á ↩
   ?(vå0‘Às?6ùÃ?Ù§?0!5B$\yen$?Ïãð??ô*?MM*~Ù1Ñ?.?5Bb¡?a?Ý  ¦¡@NJ
I©Á(J6?;§?ÿ$^3$T©"DhåÌÙ=S?????R$??ª.®:w? ?&£V?F
LÔÉ??"X$\times$æK\BQâ_!îØC???>?Í1  î.??:Âlß?|ÌCIrèØP?;©Îä?$\times$¨(ü5SCó1  Ã? ↩
   LÑPEÃ?|@Î?ö
```

```
[8?Nðª[S®$\times$*C?ê¼$\mathrm{\mu}$;¤?ãPÐÍ"(}?@"=FÆéÓgH§’’èFOªDhG?ºÉÉÄÄÌ  qtxR6è¡ ↩
    ?x¶Î
iD8x?6‘ÂVH?*XBgÜª^UPñ(D¡?¯Û?t??¡8Ï?EÐÄ??=|?ÓØ ?w1BF?=B?àr¯ßºèÐÉ??%<?$^1$ÏS?íW$\ ↩
    times$õ?Þ~cáoPQKÉ?¯S¨ÐiHM0Ö¼?Q0À/?"@B4h?>xPÅT?4¤M?@&Ïà?xCX7¨Å$?#G5?·4È?Ú#?J\ ↩
    textdegree{}ÆÄ?§Z$F¦P?$ò:$?ô???ìÊlyÅ?/Ò(’úhô?¯F=!\ensuremath{\pm}?ùÐæÏ??,p¼tô   ↩
    T$^2$ÍÄMZÉa$^2$??§O*pÕ~ðú?(FÁ«NO?ù
äJè?$\yen$LÉÈòH?¸??&?Ôê?jÙ´?O?ä
«Æ?½ðÔÄò¤D~ð#pà/~\ensuremath{\pm}‘?~$\times$¯??È¸x?¦x$^3$R $\yen$?ûÑ4xè3æT?,b5
¼tá£F?ôôæ~5Tõö$^1$?Â!¡ïW$\times$ê!Ât&@nà¨_?Ô?pmÓ7G(?w¨?s¦PBhxý?e>P¡!0ð;¤ÈÜ<6Zx?èB ↩
    %&?XÇß?6T?A0Au!<?Kf$\mathrm{\mu}$B?
F1??§ªtaHd$^2$jp¸h$Fða©?$º?(cU$\div$Câ1?~¿
iÐýß??BH9|/[?~º~Q?©?\textdegree{}’?Þ8$‘@ÑÑS¤¾?LÒ?? âÇ§ (§?Ó??\textdegree{}??Àø??$\ ↩
    mathrm{\mu}$/è2áò?©$\div$p$\mathrm{\mu}${??õ’?Ê\ôB>ùú$^2$1OÉ\ensuremath{\lnot}: ↩
    pPq>T6ÎK¨Lc?:mK?4Qe8@¯-?Á
á?ª$?é?;uJ?üCB??§E?F£TKÐ>GÎ?AQ¦?þ~iÏ5§‘B??e?‘(!&??
EJ/çö?0ßDy?úp¤ì?QR½YÉÃ¨Dz
?£‘??Tü??È{TÕÀ_???bf$\mathrm{\mu}$&¸?ÈÁ\textdegree{}=¤rÂB?!
?0ª ß’¾ù¸x"~
?É
?u:??Wõ9Â§S?/?S:T
$\times$$\times$¤?,?ÄÝF§G? î?´?«V¤??‘ü¨O=Á?"¨B??ã\?¨s ??®èÒyÄÜP¾y?¢\¼?äºYCWMª6p’? ↩
    S©X?0I(?V?K??b??A?q8~Þ
 ?v$?!?ÃZ???+??ä ÀÃ?¶xý‘Ù5ÂA??($\yen$?È?Êd?ì?ø@~KÄ?1éÓV?$\mathrm{\mu}$Õ4T$?? ↩
     zdmjRßÁ£WÃ?ýPÿr;KÄÃ5ñæÊõïR¾½Óð?2ÃM?daàÕd\textdegree{}ééà«a|AËgº?C??©Y©Y,~???? ↩
     n´à/QDcJ~PAB*È?¨5rBª   ~?b\ensuremath{\pm}1??N??
ß$\yen$Ê?\ensuremath{\pm}//?""ÆP??Ä?E oB  ??????Ùá,P=©X?z
O? ??¡3¦§ØÆ
?@M,¨únÆ ‘A4n?>ò~®?§H‘Ü?¦Á$\yen$|Fô$^1$Is$\div$¡?l’0‘5??¢sÐbÇD?®?7//.?T£?¡Ó_¾?è: ↩
    ¢ò@$\times$T3öÄÈOø>?:J*1ÎI\?6ÂàÜ
K\ensuremath{\lnot}lF Åg?E  beN?<p?Piaö¸?
{)BVvªTý£_C
t?Ñ?{oOaÌ?4??$\times$\4$^3$ ?@øEÈ?âÝ?¨?\textdegree{}?z?G¤å
5??r>!¢é‘??.ZÕáÆ$\yen$?3å=?j$\yen$By1
W?
P?6jLpEUSÔ**?ààökÊ‘Th??!lCú?P¿+3??E?JFÚ!z*ïhðaãÁ+ó ºÁ3ÀÖØ?Ý9$äÑ£zÝ»F+$\times$ó8Ün
-R?(??YÑô¿??*?~¡?¡
è[4 g*´???\?d (©6qTÕ«???kH)Ú´?PBÒ9dÈ£áPr®y3$\times$\textdegree{}ò?GSÅ?dL¯$\mathrm ↩
    {\mu}$<ns?OÀ?e$\yen$1_Ì?½{?5m??
Ôð§®H(?ÄHO?a?f@ Ù??éãå???½?Ñ!bÈ??*"^’.~ñEÔ(?$^3$N?òRz
U%!??y?¢  þ6?X?½FJÉÕ¦!rÐ3Nc®mÜ·{??’????¤)¨$\yen$§?ã?c?¢BÖAaæË?C$^3$Ô?\?ÀÁ«À=?P4&?} ↩
    qUDÐÔ¦?À@lXÑÐ??Ó1ÓV<n
»FÛ·rùá?sMX &?»w.C9hÕÉ©õ??C?mIÉð"~$\ÄB?
?ìjá5&|B®‘A©È?B?Tù
?¸MR6Ôì2~wTøC(?D/Ê?x-¡àSØTÃ£G@\ensuremath{\pm}8$\div$öíÊV?¼1?«VË’ >rÜ]z\textdegree ↩
    {}~ëJOÄý"&????0aâ(V?L©¤J£äN©£ÁQ80HÞ0=S??¤ÓÆJqè(¼Åà??a_T¡?¢h]
a??\ensuremath{\pm}Lç¶íÛÀD\textdegree{}t$^2$dåA,,Y??$\div$B?VÅy¢0
WV?Txï?Ì§ªªlZÉ?t??èê?PË??‘$& )Ï$^2$?\ensuremath{\lnot}?jøA@?Hð?8?ÏhÙ??ôPU$\ ↩
    times$½·snbibÅ)Ó(ÔåY‘AªØ??[’NJ2umj$\times$"bECî.?DH[?$^2$=?§§?z!??Ä3 aK$^3$$\ ↩
    times$??¢k$\mathrm{\mu}$:çÎ?SPjW^ª*{?ÈâK#??º$\yen$RaË$¦¯I?&?½yÊ90’hÝ~?bÿs?\ ↩
    ensuremath{\pm}*9S©?Ån^ Áò7hø§È¸ªö*:Ü??&|¡fÇ\Cé?ÑA$^1$Q?\textdegree{}xàg~)T?x$\ ↩
    yen$z?ý©Ï«è‘
ð?ô?«õlr·!?Æ§N?ã@??‘É?\ensuremath{\pm}jÑ¸IcH  =rÕ½$^1$Pð!ý«È??:$\yen$?n?hÙ«Ñ?D) ↩
    ÂKÅ¦aÃK?‘àb)?PÁ!qå£>Á?H?GìF/ãs?8 F?94/?1$#Æ«·xñ?áÌ!5?üûB?nåw¨Ñ§Ûí?æuL?W%¢  $\ ↩
    mathrm{\mu}$!:DG¡,KÉ\n()©$\times$??5?& Ës?¾#;j??$´QV‘\ensuremath{\pm}?Lãq? ↩
    d7jÝ«Çá2F!18»sÀÝË$\times$íÞ~??ÙÕ?>¦ÄzÂiP?H?å"Fø?4?$^3$$^3$¤Ä$\yen$¢?èÀ~}á$\ ↩
    yen$Åv? è3R¼º*Ì*x?|!@Ø¸MPLÜÛÀ????v8fÜáÊÛJ£Q‘?H?\ensuremath{\pm}’ºÆ\ensuremath{\ ↩
    lnot}?¢%2ËgV\ensuremath{\lnot}O&P?Â?$^1$ÒÉ4pb,Hp?WèÏ?rþ5?¦ª¤.’È?ð??)ó^(8???rb!À ↩
    ??X{vAù¢G@@NA?)}[Få??$\times$K?$?'?$\yen$lí???è3SÆ/?vÐ§4Dò4IäÍ
Ô#
*Ïsí3<?_¸?|w@2Y?? â"1?¶Äû\ST?$\div$]A:xW' ÓÆ^¼«GDHÓÂÔR¤ñKZ$\mathrm{\mu}$:kÌJ>ÂöÜÀ| ↩
    JC1????<5??Î1?~??¸,g???âF~#>Â?æ"B&\ensuremath{\pm}Ñ ü~d?5f¸«õ??jkØ´?W80ùdèÏ¼$^2 ↩
    $Maì9j7¶í90d\ÀÃRa;&?N~0âAß??â$^3$?5#íbÑ½$^1$M??
???ÔÁ!??
??ÄdB~?ÊÊRá?ïmÓ(?T¨RÔÃ?Õë$\div$WðøÄå$\times$\ensuremath{\pm} bR?ÈÕ?}
Ç\textdegree{}Æ.FÜL5:!/çá ¯?ú[ÑÉ¢DcØ=´!¡7r?qCØÉ1¸$^3$*?P?\textdegree{}2EP \Þ?GR\ ↩
    ensuremath{\pm}Ó$\div$@???ëêÊ¯ÄòÇ~åKé ÄÃªØ$^1$*)É?@ð©^R$V
 -(læ?0Ã? ?æ®«æË?IÉÁûp§Õ$\mathrm{\mu}$n$\div$'´?zQL?øIÚÅ«?<?C?@í£þp5D~!nR},ãùV?8b ↩
    ? ?½?/[W~~CrA¤V$\div$Ý'??Fï?£'ì\textdegree{}ö?é*$\mathrm{\mu}$c< Ë6¤IÄÉKTðJ? ↩
    ÜO´ùÚ^?l\textdegree{}?ÈÐ5lÐú¿?)?¸Ñ~??ÇîH¼Ú??@õ??Q??õå$^1$r?4/¶Ýú%¡??X$´T"?}[G/ ↩
    DnÜ|¢KdÍ6LÀ?<?ñ bf9ÐHÔäÑ¡i?ÁaÌZ$^1$?¾%d°c':#!wr7(?á«=?"âØ~?a Ï?kt?Â??bEBä$\ ↩
    div$,äEÒÝ¡?ÂrÉÉ??j9?-Úýj@@pIða~°ÁdE?UmÅ?z#Bõ$\yen$Zö'~ÃtzÜþ?0åV?Í??ûväåZ$^1$|ñ ↩
    ">À)vàÃ?Û¿BM?ÀL¦ÔþHÁ?/[·#]?1{òá?F??<»â(úâJJL?Ô?Ê?Áó7<
cç,0Õ¢?Ì|å$^1$#£;N^![;ÿ$åÏ?X?´7"?7â/?4ûÓ?35<Ó_pdå$Ñ[$^1$k?Ûæ7~T9Ô?ËJ»|¤?ãWÐU4?à? ↩
    sSÜ«ébÐLé?úõÏk´ÜP:·Ñ?=?6Üª$\mathrm{\mu}$???j Ç¢4A?c???$^3$Æã?f??a??r?Ü??$\yen$$ ↩
    ^1$?? ÂDà\textdegree{}ç6ðnarñ?ì:tò\?ô<?9½$^1$tM"Y Ó*cýp ^è/aÉ?ó?Ã?¶
D'?¡B?!Ãcnø?ù
9\textdegree{}?1!Dxk#Ê·%w?ùÄ~ËI$^1$7A'Àì\ensuremath{\pm}?¡?'QOãímÑ?L§Ü'\ensuremath ↩
    {\pm}9Ç?ÿ´ÝÏj¶´0\textdegree{}ó*êAó¡28ØàNè[´ô?Qè0ÛJ(¤É?«):>·¯?å¡?é>?Á,??$\times$ ↩
    %91?¡? ?òaï??'~Å%$^2$ÆH3??âs~QJð QÑÆ?Ä
ëjjkOÊºÏÄ©oz@p
:?$\yen$å5ÏH?Ü]°2ëU?ôcò¤@£Vjý?41%"_£??/\9G&?iÕ¤êÄ¯?b' ÎJ)^J\ensuremath{\pm}$\ ↩
    yen$HÜ?8?R?¿Ä^Þôqì'åmÞ7í?-{{zðóø\i\textdegree{}0ããÄäN£W¦??p??)åÞÛÛ'Dâ¾Oô?\ ↩
    ensuremath{\lnot}Jûr·t©Âjü% ÔÆ/?$^1$?!R??éR,Ü$^1$! ~È?øQæ?y~!_)+ÒL?2ü£¾¤<Ñýhò?? ↩
    A?°2?$\times$DÆ'É¡bÂ?~ÿêÏ$\yen$å$^1$0áFàeéÊ¢x?2?
©?#QÎú|E?ÁðbÀÛ?ôù$^2$S32&bN8Á?íÂ  SR¾¦$\mathrm{\mu}$¸VD??><àm|ú?ÂÊAAÈ?,?NXpìCS ↩
    ?#'7?Èñ~~õgØ?|Ø?J3?*{Çô«?rð§6y?8âªR´hA»OBRz?¤Xt¦?7´?Ç'$ÔdN*ëWÔñøÿè8rÇÆª$OòPf!? ↩
    ÚUF@[pÀGgÐ$^3$nX.??9](C>á?põ??Î¦§´?0U2PªDçzþ&£»V0p^j©ãþú\ensuremath{\pm}?Aö£B' ↩
    Ix9x?ClKq¡èAô)FºDÏA\Áã?r?¨ú·ì§D?dö?øN{ôåS}ñ«'hUÉ#ÕêòE$ ÿh¦?ó^!8?+vý¨ã$Ó¦M?q$\ ↩
    yen$õq$\times$Öb? IÁ%åi¿üÎQI?jR$}-^=4WýA?Ã3\/Üfà»ñT4Uösh9@#VU§4Åõ½~»~cf«<1»]¤qÁ ↩
    \textdegree{}i??]R=¢|$T??ùÔÛÀS«ÿ½\Ì?°ÚHÂLyå»Ñ?k((?Û+Çt[óãD¨?O?V?t$^2$Ê
?5=?$\yen$ÉAÏA?n]\éÁ$\div$?ýæ^öaçç??·|>m?üÀ+'4??¾M/B?màG?|'òy©(ô??E?vLs"2+MçV@ « ↩
    Ûr§´V?,#b¼ñ¾
?w?&?Ày?-ëj'$^1$Z$^2$t~'?}h3ÄúIXaêC¼?$\yen$íohò>?qíÏæ?5?HPÝ2D\ensuremath{\lnot}D? ↩
    yì'?õð´\ensuremath{\lnot}NS4õ£v<Î?AùÚ$^2$}&íÛ¿!??¸óÀùÞA=??ÇãñÄ"AB5âA??ø&?¿x{Õí ↩
    ??u%?._ 281ÃÃä???a@îNiòW«NNYýkÒm«àûX(8ép"(P@]$VçÑ?WV\$\yen$<ÎÕ_i¸?? À?i?ÄG#- ↩
    ¢dYVT89nH0ÊºS\times$7\textdegree{}
ùãø+tfdÉ?+Ñõ>K?7¤ÁÌÖ¯â?¤¼áF5Ëcd(?zTG$^1$?WüÜªúB#$^1$H£$^2$8ô?:ÒÓ£??öi@îJÅ$^2$x?Ä?? ↩
    LGH·H?\ensuremath{\pm}Mp¢©'[¤Ò?&@äLí\=?^E$ÉËÔa?«
HP?ÌÆ?Ôj>Û1UÀ*h7½
Ü?ê??|AÀ??snúö?u?¯2??õ'úÔ
?2~?4??K? È )2·Blt¤¢XèBÈL>$\times$k\textdegree{}?m?^»¯]ÛtêçÃ   35?<(ÈÈÈõ*?)OÍf"a? ↩
    BdxCR´´´?Éu???D?1£FlDD$$D@hBÑ-&v??9?Ëýz??Æë´®hÒ¸{v¶·^?È$K l<"???/.\?®ív:ræä¸Û?à ↩
    ?A?C I "I?ãc"aÁ{{áyròæåâî??2?hh$^3$QI(G/>21?$\times$ô??É£A{ysÛ.Æ?ÓkKG.º¸$^1$°ðÝ ↩
    ^¾ÅÍ"Í?d4?CåâË¡«]'Þ^Þàéù àyGla$^1$É\ensuremath{\pm}DDC?-ÿ???úð\ensuremath{\pm} ↩
    r2? .$hüÌd41+0ñ?Eæ?ecÄÄ{(~)FðÞ^¾*ø?? bãba$\div$\LLLlþÿâ?éý?õ?F@õ2??Î?óÑüù??¤$ ? ↩
    L'eÿÌ|l+ìËÜÎ?¸¸$^1$?1$^2$!_Dcy!Äybð!á\ensuremath{\pm}!\textdegree{}ÌÂ7????Ë? ↩
    â¾ÃÆËÂ
  ·Æãå$^2$øñ\ensuremath{\pm}Qxñù?¼DQ0ãEÂDùâßçÄE½ÍbÅdEå¢"£ñd~f$^2$Ò1H£sy?Ø}

?&f6??,noÍ¢ñÈ??ÀKfZ&?Ï,kìçØ@Î¡a\ensuremath{\pm}bñb"1ã2 H?ì)\ensuremath{\pm}&Çe$^3 ↩
    $xÑð\ensuremath{\pm}ðææ$^2$æ??Îf9\textdegree{}ØÆøQ¾òÅ_¡¾Âñb]ùð|p?ùïð/b<&' ↩
    caÃáB_ÆâDe\textdegree{}$^3$~)WyÜ¾*O9\ensuremath{\pm}QøÉrÙÜfF~,Ù?ê?yd9 ?¸$\ ↩
    div$1Î¼ý$\div$$\div$^?»¼.õýÃÏ?Ùýæ
?ÈÏ?g"?)?Y?vb=<blþi^s<DT$¦rQ&w5ÏÆ$óÈávo???Ã|úÏÈ$\div$Ç?¿?ê;?Ak?ç5ß¶½ÈFl!áL1H¯Ó5vJi ↩
    (¼êCb\textdegree{}
ÁÁ¶¸ðÛd;|ï
¾?ÔÌ'ä¦?o?íô?[Ólä
ñ+ÖúR;
?)Ë¦ÐúJÕN+w?#ß¸Ç}ãââq-+Í$\yen$eaÙÄoòBG¾Ë?qVE??¡??ÛÉæË Î+òh¤v{Ï_Ö6Su;E?á¸Oí?\ ↩
    textdegree{}¤üÜÕö*Y$^2$æ¾ë'dfb!#ÁQM$\yen$ôÒEÒãMÅÂÃEp$^3$Isé$\yen$(túßm-?Ëm?£$\ ↩
    times$ÌH"ïcÜç«{tÁ?·ÀçÓ»»e$\yen$zË9Y?\ensuremath{\lnot}ØéÝ?ï?Úõ*-??É$^2$*?yÓd
Üîô+RÐF«õ7LÏ$^1$åÓÕÙÓÚÛuù®$^1$7?U<8ÇX^ek|?\textdegree{}æ=ìkgGío1?=þ?yâ^_o3Ã7=!??Ñf ↩
    ]«ËtÇ?Õ?ñïuë¡¾¾':?kê¤ô??¶È¨§TÊÍ|?Û¾änu@û?Ý¶Ãñn? ç7?/?Óa{?üëÈÙ?ï¯ÇäjÚæÕ2½^?å-¦Õ^ ↩
    uÛÜ_25;4?¾Jì+9Ø?]Â$^1$T½7òÖ[Z,$^3$sØÖCf»¾¾4Ûç\textdegree{}-\ensuremath{\pm}?ß$\ ↩
    div$ìk¯*9ù¤·7$\div$P®~+ÕáÇ¦$\mathrm{\mu}$H®aN'íß-ÙÇ;çf¼Zÿá¶ûïhvóY¼<°èêÌÂ6ö¼Fû
æ!%c
ú¾\ensuremath{\lnot}G½îw?!ör?Ë9§Ó>6ý6íì-.àíÓ;9E*õï[n>)ªo?$\yen$øcyÚ0OÐ\ensuremath ↩
    {\pm}ÈâÜÖ:ðÔqZp/cë¯z\ensuremath{\lnot}ç$^3$Jâ\ensuremath{\pm}üùÿTv£. ↩
    RtWÕ8îmÚÍAÒV§O¢;Öî$\yen$[ç¸$^1$áýÒMy?ã\k?úÊ¿$\yen$§?ýÎ?£»?ÜIÎº?º?{wkI¶9mâSfþ%ÍÍ ↩
    ;gLrO\íåmÒ?Ý?$\times$e}cºËÍÒèsl>$\times$o+¼vWóyÞòwúüÛ]*Y¼^ãÍ1Û·C¸wÉ{¿~;Ýðuæ;Uò ↩
    {1?$^3$$\mathrm{\mu}$É4$\div$¶´3ß¸ë?Eg}áóÿZÇvÏ+¾WSkÞs~c?4n~?ÕÞðêöyéú:<ÙGQÏÿÕ? ↩
    ãõýoYn$Ê6/ä\]1ß|ÖÏx?;&_O%ÕMbÑ·ódAV???½öf¼ý #·?~^síÈ]Â{§|R=éÔ'{]n$\div$c½Í$\ ↩
    div$í3sw?öû]ö?/ô}gñé:eJ¾ó æÂÀ¶Ð\ensuremath{\lnot}:EÏëFÊ%?î$^2$$^1$?ÞÒã¨ÎÜÐ®Þï~9 ↩
    ô¸K$",.moCú?M$\div$ÍÁþß=#?G5?Ê»<½¤<LVGïM5êdHìü-(ôæçð£hó®;mº8íß
?$\div$Ø$\times$C-Öæø=û¨\?mþþ£ó|ßÇ}ó|Ý6TÜ2gÙ¶6â'E?(?qÒõ5^Ö)·GJÞmN¢;øh(ùw4FÆiww½À¤ü ↩
    ?ý4ÏøüßŽÿËéùº,ÔuéôðuR?RØù]H¶én{oÿT?$^3$gió¿»eé&ø=\ensuremath{\lnot}áÔ?¨¶|$\ ↩
    yen$$\times$Ý<Cù:îÒq)>~W?çû©??êe?Þ¯¦oçÌ?'^$^3$-wæ;u?àÝiÚÜê*)$\yen$¾þ$\div$1ñ? ↩
    îÌëèä($\times$H[l$\times$g%ÐÙøð©úü>^ÿíâþ?=m#??ì:¶¯î_â\b?¶K$^3$ÙÒûøJætï¾?Qm?Ùu$ ↩
    ^3$»ÄT~$\times$¯7$^2$õh¶n$\yen$ÓM>¸¯$^1$ëõ\ensuremath{\lnot}ø?Gw$\div$??$^1 ↩
    $6»Yq?
Ýè$^1$ïÄg:t©#ßè?ÊkV$\mathrm{\mu}$.$\times$ã~[7Aö?øöK½ÅÊáâf3Þ\làCÂòpy¿?íÔÚÒgråþ?$ ↩
    ^3$?Q~LÓ?3Í7c  #$\mathrm{\mu}$8ïO_ê+þ?$^2$?fï?i$^3$jçÕ¨ÁÁíÂÄ<\ensuremath{\pm}$\ ↩
    mathrm{\mu}$_Èò$^2$$^2$½¨©bÎRÓÁMSãô®d´?öU¿wUàÑlõ?$^3$§]§W\textdegree{}Zn?I^9D$ ↩
    ^2$ $\times$z=¦b"?89$^2$?Çøí·ù?Üg9?ü{g6½+¸w?®c½]?xí(/?c?\ÜºÝî}&Ös?5??Ë??&??º¸ø\ ↩
    ensuremath{\lnot}?\öõôf$^3$D¶ÎfE¡¦ßß]}¨?]ó\]Û$7KK¼õ6V®Ì-ò»À©\textdegree{}{ ↩
    câÇÍçqûÆø?èâÓ??ßKce^ÛYºæ}?ö>1\ensuremath{\pm}ÖÐÉe#;ýb+NE#$^2$D¯YE[~Së;<ÿÇ½?Ñªy? ↩
    ò·ØifõI?Ïù?«$^1$»7^í¯EÆ?&Â?=ôfBÝ?eõ$\times$¯ËFè6^,¾6]V$^3$?ê¡üD-ÌSÊ?p«?KEÞ$\ ↩
    yen$ógtëY<JûÍ'¸ØÙ?©c1m?¡w?ÝÃ$\mathrm{\mu}$O\textdegree{}}k?MucÞ¼´®¸Íävu?å~I¯-? ↩
    Õ?}\FÛäOçËÄJÆy¼:ìD_?EÙðØIâÞ;Å¨s??Y?ë?ÉQÏñÜäö8$\times$?«$\mathrm{\mu}$q´S?¸^ZUfÇ ↩
    ?¿Ü&[5!§¡ÎçáP$^1$]?~òâ¡sáÂò1¦ïõPpYÚÅó>ÞûÖ^ÅìCUj,?ÞL&Wt[E\v{Y@ZqÓ¸p©WM¿ÇÑeÚyo]$ ↩
    ^3$âò¯íª?bâÛr,ßÚïøN#?e¡{&$^1$B¶nþbm^?g·F$^1$BÏpã¦ý¨Üþié\ensuremath{\lnot}å\ ↩
    ensuremath{\pm}??Ã?Ïèáw/m*xýýý%¯&¿wÄw?yÍw3+ró=78ö+jw$^1$$^2$£8ïß$\mathrm{\mu}$Þ ↩
    .ÝìäÓÉOº7D?9YÇÝóÒp¸¶?gCÁ$\times$âÙùüèr:ß^?Û\ensuremath{\pm}Ø+Ó-;ÞÖëý\ ↩
    ensuremath{\pm}íí?Q+Ø??¨õ#ì´Z?Ç|ß·qaÒàS8eÒû:?ý^?AåeÏ
.>?g?·??EÓøäl?$\times$ô$\div$»ßnãÔøfy$\yen$ÍFÛ1oóðøU¼ú[?ãf],;?£»8«ø,TâswØÒK?:; ↩
    ØÒ¤Ùw\ensuremath{\pm}1ðíËñ+¢Ûû5pÚ5~O?w?«Ñ®gÄäý?=??|kNK?+þuc?ß¿ÿY?kÛXd;ÅÓÒ~ó¨$\ ↩
    div$^Ñß'©F¶wë(K(ºJÎ#Ás¾LÍ½?öò]ôyîº$\mathrm{\mu}$?Êã{´8«|Ö$\times$Õçæ"¾3¿zÝEw$^3 ↩
    $ï{þÑß*»Ý|Ö?Æ£7?MW?¿útû?.?'m¯[Àçý$^1$?W½o·øÖ@KHÎEM?Á?7ü?¢W»8í¾ÛÖ¢Þ?F¿g7îùb<Ö_$ ↩
    \times$I¾aaMÉäþ?Õ5/~Ëú2$^2$à$^2$éïø?ÉaÙ]ÅiQh¨Îø$\div$G|'?]$^1$;l§k)??Wáã/ ↩
    Ê¢ºÑ¦þÜ{??Çå{ëî¶¯ç»¶Àm$\times$yeh$\yen$~
#¨¿pö>S¨¨¿vÊsòeôY?}Û*Û®?#PÓÉ¯èÜ$\times$q_f-7ý=ÿ»ÁÇºk~káÑÉ.Ú?Vað?ñQ®Ü®;ØÕMÐÍÞBw< ↩
    ¼n¯?ïä$^1$¤ßrz½?·}sïÿ:Oþ¦{-¾®ÎBß5ü>$^2$j»n?¯wD?¿v·NqÓ0^è4?Ö¶\ensuremath{\lnot}º ↩
    |?_?¦çI$\div$ÔÂg¿·Ï?}öýÜ(Ï?/§\ensuremath{\pm}>ºÇ?@·W$W®ø§¾?ïê?îç]¸[?5?åË9$^2 ↩
    $waeÍì2a\òÿBßþÏõ¾ÁwaIOà7$\yen$ô|ÜùÍ5t.áÓQ$\mathrm{\mu}$?Så?OªwîÓî(¶:uÜ???¡©ø ↩
    ??&VWÕÇ§ó$\yen$æÿ[¨Î- ß"ê'$^3$ô$^1$«W+?My·ÝÐ?$\mathrm{\mu}$9ÛÍæôïê?ømö~õ?N~6&1/ ↩
    C¢ÍÏ;Ëõ$^2$©$\yen$Ë?$\div$ü$\div$v,]Â¸êÛx_=óÝÄJl¼Hæ$\div$2ÊÒÑû¾ÙÞ¨-.$\div$j¸íæs ↩
    ?!j§þ;?WoéZS+?$\div$»ÿ$\yen$»ºoéæ$^2$âp?wwÉÑ$\yen$ÂK7ðí¶?}Ù?G¢Õ?º;iÂÖê?ÙÖ$\div$ ↩
    /º?E_áýxTýØºMõK¿«ü&©¿ö\ensuremath{\lnot}Î3g?7£C«øwiè¦ÄPûÇQû;£½ýmÚ¼|Ä/bâ$^1$+ ↩
    Ûé$^2$úwü&?ÝüìkYTþ?e?u¿Ov5E¤v&?åØÒ?VÊûlÎ£ø?]=ðúÔKtf)Ñ??¾¸$^3$kÐèw9]?ÇÖÄëý½zÆ]?$ ↩
    \mathrm{\mu}$6,mí-6$ÂVr?¨¡òòÇmÖ®$\div$5?<$^2$PË£Ù%wY|$\mathrm{\mu}$bÁ¯ßóö??ÏçNõ ↩
    ?çJ¤Ün\ensuremath{\lnot}NwÅå[ôés$\mathrm{\mu}$?QPö}yù½¦fhØ?½???wÃ[ÇàX$\ ↩
    times$uZ\textdegree{}z¡Mõâ]_s^vðçØàGÇÃÌ=ÎG.ÖÊd%$\div$ÿqÓëUþíîæKM?35??6 ↩
    £ÉãsÞÄÄ¨eN¾¶ÃùÛ7ÉÀûüwvÖwñVÖÎï$^2$ê¡\ensuremath{\pm}àj&vÙÌ\ã½½??túMÑÛsquSþ\$\ ↩
    mathrm{\mu}$ÿVÓÇaÆgÍáSWón?ðþªf??þ_W¸ìÈi?Ðê1$^2$þ$\div$$^3$~]·ùN]ZwÉ¸V»sD.lÔ# ↩
    °îöñ<U$\mathrm{\mu}$¾½õ/:ÂÅ}g7Ima_fïñâøVø;Ñ(ÅÉEí5Jd$\yen$=$^3$¾M?Ùß-?ñ^Öw?íj¶ý ↩
    \(Ìª Ypu9??nØï¸«ñï?o¸7Yc{tjXì?DSÞ¾O*qß?ÎÚÞ?$\div$½$^1$ùí?nvÿ»?ãî2\ensuremath{\ ↩
    pm}_Èát«lª$\div$Ö.m>ªjd}mÿñëxaTFõbìea?&¡[0$^3$Úø>S¨?ûcÛÐÆNëGÜÄÑ"ï2ó4\ensuremath ↩
    {\pm}¤àStðiù|åõ.º??ìJJwNì?Â¦agæõ?b$\div$ü?Úz
5?ú½ïî;^âÝÍÅ'Vn<mí*kKÎ#?V?wYÚöÖÞ?êËêÃ?Î?-Î.?¦r?ÆÙ]?ìüÌ¢´~?öûìõÊ$¶ÕiMÏHãÚx¯\ ↩
    ensuremath{\lnot}vÊÊ®»Â»\ensuremath{\lnot}cZÍüL??$^1$l?Øs??¼öP´{?]>oP"/g1\ ↩
    textdegree{}Ó$O´ÖÊÃbft$^1$vÎoº6æ$^1$Õukln¯^®ÿïçòÞaVÔöÕßø¢?Ébá&O\ensuremath{\ ↩
    lnot}ÚÍj·
Õîý$\yen$»?íÍb$\yen$¨å?fôfãÝ]Y?/Û??V=fÕõ,]$^2$åo$^1$þX,®$\yen$;Øæ+Db] ?çT¸í$\yen$? ↩
    év»þèu
eóÐ¾??oAÅ[>·|îZöU?+n?Ùu\ensuremath{\lnot}]ã;k?$^3$7a?{?Ä´:m¼Â]Éß¾ÖØ$^3$$\ ↩
    times$jKúë2?ÄùìüVL/¨ê´ûìiz}*¾?àÅâ$^1$åÝ9ò_®È$^3$Úf<?óÛ
ZÓ=íÏiðí½Ãm(¶Sû¤ã~´¸vúÅbÂ·¿Î£éûú??úÎgï?Á©b?+õ!odw?zÄ$^2$ôgOm(w&ëgêjéVm¢7??ÆÊÄù$^3$ ↩
    .?$\mathrm{\mu}$pÛü».Yý?Ê½}F$\times$?-ýÆnfõ? ¦ $\div$õþ¦ãÚÞNú»?:?©gehr"<jï?·dæú? ↩
    ÍË*·vßNûÍM¿¨eÙ$^2$í$\div$;x?öxþt·?tI#V.Ûê7ZïwmE¦yô~ìÞâV/N$!ô_w·àmkßðÙÛ<¨áõü$^1 ↩
    $Ó«NÓawéWÅI%¾}$\times$[¦W$\yen$Ök$\times$k½ú%©f¨$\mathrm{\mu}$u?ØyXÛø$\mathrm{\ ↩
    mu}$¯,\ensuremath{\pm}¼ÖË®Ûúÿ¢¢?ïÀ©nÊÇeØVer?T^bß!qÚÙ¨ÅªöÞ®ÆkO«ôÕ¤J$^3$9n??? ↩
    ·_yäk
?Ñso?{òÂ$mmåëøò6ÍÝdgßÄâUçÖê"ôZ5´??]=?ç=?DÕÓhò?Äyðäôì^ÛaÜóY>\ensuremath{\pm}únÚÀ\ ↩
    textdegree{}eõÙ¯èYÄÄÅ'»Òùu?)®Í?«äM1G¶]D¶o;%ï(QÌ#?wåÆ$^1$'Æ$^3$¸ËÉÕë]a}FÕí$^1$?ü ↩
    ]ûhD»ÌU¦Ì$\yen$¸ÚÑü:/WMíêèe?nÐKÎâéÑç»ø?X~nEvéÝ.
e?+ð98ìiø?ªmk®òÑs$\yen$^.W$\yen$åh}ÚxÝt¾Ó_¼8?"ì4ë¦¾5QÁ$^1$lÉþ^?N??ªæ£èà$\times$; ↩
    eÇq#êçrØ¾´æi$\yen$?î:uv¿cB»V$\mathrm{\mu}$v¼í-?W%ß{©$\yen$\textdegree{}Âº¾_Ë\ ↩
    textdegree{};Ä®þÜ?/û?%ec>m_~î&ú1lÿÞÇ[ìk&$^3$?sÞÕ{4?NÜIy?J«ÅzA?Ó?#aYkkàóÆÖw)ðlk ↩
    ?'\ensuremath{\pm}SgæÛÖ¾_WW?>ºgC®7oëmèvË·§kd§'t;\?É)J·? x |8UÎÙâ]X}®/X}?ì;Ü%ý?Ù? ↩
    Ë?
?nE~?$^2$q??§ùuºý?¿Qïú«sÆúYù¿*½?O#?e§Jõ$\yen$·®£ÊÉÿa_JÊ"2kÓ¸í$\div$h§3ìÍÔ?ÏÛÐMè4? ↩
    YVÛXF63A??'«æ[?khÿÃ{ÞoåE_Çúc,Ùpë;?êmfõ$\div$¶V»Éå]ßVos@£U\textdegree{}£Õ|G?¢ù5s ↩
    ?åaõ?twÝÈKX$\times$Õ¼èkk?>_2N??"§ª?¿¯{®¿çî}Ý¯7U§;_E=îït{_çJøä'Ìi$^3$$\times$? ↩
    ç¶Ø?~$\mathrm{\mu}$+65ü
ðÜ??:æ¦£Õ¶¯´}{$\yen$Wâøã e?Mm5z)ê?3ÜQ®$^3$?9"!ÃÄ$\mathrm{\mu}$¾?V\¾?Úß$^2$Î?oùñ?| ↩
    î¶ý$\mathrm{\mu}$|?ã{XÝ»úßyv$^3$«)æ$\div$¦ý½2OPíW»¸Ëh;"cZ¦?ÓÇ»¿ÈíúZùÅ%-cú_\ ↩
    ensuremath{\pm}??Þ'*æÁ\-ZÜå[óÂòú[c½yVvÔílDéÇkté$^2$?ª4Õx5é.y¿Öîíc¿eÃe¿yMÐß$\ ↩
    mathrm{\mu}$¾Ëvy\ensuremath{\lnot}oâôºøúHú3EiEª{tëvk~?¦|ZÓ¾?SÉ*ÓFÊHhkî2ò?7X| ↩
    êqÝFûî»íý{æ?þ"»#+¯ÎÓ'®¢ÀJGðîý4ZmÄê?ÝÙÛ?«t?¾(´\ensuremath{\pm}[_x i ?ýÜÿ?5 ↩
    ûïÇ§¾þ¼ªfN9vnF&ñÙiNô¡ã½m?ÒÇ+ÿ£¶ÞÆÕlg$\yen$\ensuremath{\lnot}Ó%Â»6Ñ[!½G? ↩
    ®ÖºÂù3ìÝÓÝÓXÔ;´î1îãf5YØÙE$^1$$\mathrm{\mu}$z]évpºÛÐë½ý¤Ï§(¢{1?'È{$\div$}Ø?nê[$\ ↩
    times$vúïÙUõVõtù¯\ensuremath{\lnot}¯ì¢n""ô?$$^3$Ù¯Wg§Ô,\«NnÁ??I;Ý????¼???ó""¢? ↩
    ç2ª¯yÌ\ensuremath{\lnot}ªËßUMT|?ZÚÚÏÃë{Ûm\ensuremath{\pm}òZ¶ßù{m$\mathrm{\mu} ↩
    $¶·$^1$ÇUª$\mathrm{\mu}$^ãLÌÌþàÊÂª>?DDG???w»ÿxw?u¶DgäóÃ=Ird
áæ¨M7zbVë7Øá??û¿[/öã^?Å$^3$?=u\textdegree{}%K¨<???H.lU«:JÓr'A{Cm??Mi;1Ikîm }ëx¤VÙ2 ↩
    ?gê-ÑìáàЗÅÏaÝ#55Í?B0?Ç\ensuremath{\pm}$^3$.nÊ¸:j\textdegree{}Â?gCÒ?o?%eæ$^3 ↩
    $6ááy?îzx?$\div$<?] c\ensuremath{\pm}{?$}^??Å?ÏZÞrù?*?+PÏMÎqyTé.SÄ»?\ensuremath ↩
    {\pm}$\div$,?IÓÕë_¿$^1$®Ë$^1$8¦ø$\yen$?©R£Ú
Þ¸+tö=.\Étfé¦ÀÛ???îC+3dtà?a¿?\^~?þ¶gÆÕþè\textdegree{}Bî¦Ø¢@¦Ï]Ö§39_F?7ð"Ç6@Ô\ ↩
    textdegree{}wãÀ/%áÍÖhàîÏ¯?_?Óýw\textdegree{}$\times$9,:Df?EuFC.?=X{f\$^1$àÐÎÅ? ↩
    h¡?¦®¸ÎÉ=A?]rË3jé¯\ensuremath{\lnot}cZ;?ÍbjTS+QË:?¨]¸ÐÙ?(~EG¾?&©é=Õ# 9 [:J?gä4 ↩
    /.O¯MÅ"'\ensuremath{\lnot}'E<8dªR?8_~
iÈ;Û©áþÞþì?\textdegree{}$<~èÕñ7OîY?lÐ?eÞ??fp¶?Ëÿ£oK
«àfß¸By< -Ó$ãÈÁ/3b?Ê?+'lØ>ÁÊZ¤¨¨iiªM6?Ú?o6cjÌc¡HË9ÖW?Á^H-$^2$ÀQ?=Lt7Ä Ñðeî¼?]}\ ↩
    textdegree{}?Ê96¨4PO~ß0LTFv¢Çgª:[ÜhN$\mathrm{\mu}$ÉGÚ78ze>?AH;¨jj$\div$´ÞoKc? ↩
    ¾Fª?P_.8Q~®¼ôÙÄA$^1$???£½9?.Ü9Ô?Í?û?è&$^3$2:9ê??ßç?>vFS/#?d@Õ5qPxÉÃtt~ÍULÙ$^2$ ↩
    /;KM!äCÑôGÒ\textdegree{}@¦??6L?@jãz§v»½UÙù¸$\div$¾,$\div$ÈO?¯h§}-MmÜø?ÊÂ$ $\ ↩
    mathrm{\mu}$?"H?íRp%? ÀRÏ?%(»:ÓõxL¨«zßVxýï?üÔÐè?Á$æ]¤r?FÆv??XêhÓ´?Ù?¸¨#ZYñÈbáó$ ↩
    \times$Ú?ëôfòVöpbl^ýÎY^õòË$\yen$~R¾ÃTNÍ_#í?R$^3$ªÝ_?ëRNp~§?eæjn:Òa¤7ïï¾ÁÐ$\ ↩
    times$iÿ:9L:ýGÂ\textdegree{}þ1?#¦&d.57'¤®ÚóêÌ¼WýÞ?J¯§à?¦¶Æ?§îdabâ$\div$ûõ~wþ^á- ↩
    ò}·¿$\div$1k&?îuçzÞ@Ë0_Þ\ensuremath{\lnot}?ò\ensuremath{\lnot}'®®G$\div$|?Á~ ↩
    TNÆðg'vf;%°;[óý{ñöõ?/Û¿àa{àò$\mathrm{\mu}$É?¿¸\D?ñ?3)ê?JËÁ$\yen$KÉ?ê?>_¡'$\ ↩
    times$?ú¯-¨§iìÎM½¾ñ$^3$ó~}\ensuremath{\pm}}ûõ:wÛÓéôû}Î¾[$\mathrm{\mu}$$^1$???pK ↩
    ?1Ç??;;´??}%§þÀ?h7Óx$^3$?{?cp7¢èú/à=_¯§úuz};ß»§}{Ö\Wk?8åe,RÖ7?é)$\div$ãsBôOà«H? ↩
    ö?O'~|+>>%J?
ãÁ5¾ýXþ¿Ý Ó?~{=î¯å{ã¾e\ensuremath{\lnot}DÄS/L??Viê@¯âÔÒ\ensuremath{\lnot}YO#7?B©bË
o¤D?Mò9n2ò=ðý$^1$¾®?k$^3$ÚÃü$^2$ò|·\ensuremath{\pm}·õW.a?Ðç?ç §¨
yP|?¤$^2$óáX$^3$Ã?Ð©þ«¢n%p-?dÊCª$\mathrm{\mu}$oÿg´Èñúý>^Ë<Íf+t$\div$?\ensuremath{\ ↩
    lnot}Þõ»@Ûg[UÜGÄ##?EU¿?.©e$\yen$¯K¢2?ýêÔ\ensuremath{\lnot}àÜ?ÿ{¯Òéy»YÎ3ü7$^2 ↩
    $I¯óîÃ.f\textdegree{}$å?$rcØ?ýón5?Ü?yOa:Q?:ËtÖôj)<&#v$\div$u¯ÖÞâù}ËYJÂÏð$\ ↩
    mathrm{\mu}$?Ã$\div$<Ú?Þtyi¦¨¸wTI©ÎJv´"i©{?w&B?n¤¤É?£ÃêÈ[Û½¾¿ÊÕÔa?«=¤Q?¯ç$\ ↩
    times$Ö?¼¡$\div$Y©$\div$jÂ?dÌ$\div$?á$\times$}Jío9tP?¦_«Ä·$\times$·¯ùö:>ÞÍéôÿ/ ↩
    Æ$\div$õq£¡éØ???$\times$??ºÞKýLl$^3$ÞMOwÅ¶èX[üË.U+æ\ensuremath{\lnot}ç[*? ↩
    òºÙ$øQ$\mathrm{\mu}$üß®'íéõÿ/\ensuremath{\lnot}d?u?¯SÇý_ääÌà?2Tr??¢$\mathrm{\ ↩
    mu}$?Q8$\times$T?<«¡g,ÿÌü/$^3$§$^2$ñJ?ò\ensuremath{\pm}]]WÃf?IÀÑ¾õàâjzo~ ↩
    ßÖû¯õôÿÛöû:¦?ï$\times$?  t8¢íB7<iUAêÕ?Ð?Å$\yen$?f'IVÛt?ûåz+?Ð
ªù·Ü¼ö$\yen$¯?z?Å$\div$r|$\times$Ë¯oZ¸Àûzx\ensuremath{\lnot}o>æ?©¢ïVø?·@Û??$^3$t(Ï ↩
    ?$\mathrm{\mu}$?àí \A8õ¼ïoilåX§E§I#|Ùr\textdegree{}§?ï\ensuremath{\pm}Æåë?m? ↩
    ®ÁØÑõãù6ÄÍw?}Æa?¨4$\mathrm{\mu}$6"NHÞñÇzYÚMá@,p?ôuôÖ7?o?È$^2$åJ?3Ëà???M8ã¶?ç-q¯ ↩
    ?YÖÄml·?~Ay¤fFÓ$^1$ÈP?ãÐZ$\div$ÛÐÅÈ|?óÌã~4-$\times$?Þý?ÅxêìÀCÅ
!?V¦Õ'
¸?ÏC&[Ex?"??ìmÝ4¯!?[|ý§?5ÞùsóvÖèº(ÝÖ?CÀ?\textdegree{}?5W)+LÃ«$\yen$~¢ "d??"X?#;'@É ↩
    ?sPÂ*?ù?¤?õÃ?ð$^2$o©ºúõäq[z?K+JY??B$^1$Tüè|kuJ)Ã¢«$\mathrm{\mu}$\ensuremath{\pm ↩
    }¦ªª¦@ZA½ø,?mÇWj?ì¦Ì???-?ÙEF?¯Ý5:?$\times$¶?£fÂ·1»¼8Ô£ÀKUì)\ensuremath{\lnot} ↩
    å?ª®ÂM=?A£?Ñ$^2$Äv?¶ÚÒ?Ç?ueÆ&ê4?Ãà?1/l
ÉT@nX¡
Èp$·{?$ºwÁ?H#nt?Ôtjlì),??@sè¨
?m?oo'3obNRÝB#fjFÑ22¡Ü!
QYçH?\textdegree{}G?«?¾ùrR%m???v?ió3åªK( Ø?
å?é?LÞÅdáæÓfó3J?ü¸?ßÜ?R:%$(@¦dq¢  Ü2FÄ-@ATÑà?ª,DùÓG'
£OA7??K7z;{}Ã?{¸Þ¸ñ5\ensuremath{\pm}¯õLGÚ&?HF½EmÐ!?
"ZDi)&ÞËtIå·ú?$\mathrm{\mu}$\ensuremath{\pm}Z¯?gI5ERV)êTÓÑ3[P¢eî=Ðz[i\textdegree ↩
    {}?/49Aø$?|Ê$^1$??£æL[?'?!ëFp¿eþã?Õd~T§?ñ9©?¶?:J????RCN?+ô"l'xòsß´Ôu?Äy?I?Õ?ø?A ↩
    &A:©ô?G¡'$ð;{N?Y70 x%ç?Q???ú9$^2$ÈR?p?7¶À???lV0>[¯ öepÆIäE§JM?&Möð?¦Ë?êR\ ↩
    textdegree{}!>:á07c*?
Q??y ?I??Ö?Xä"k¯jÀ]iÖ[$\yen$àìio?5.m?"?£?õ¸HÚº]?FÀw ù$;ìX½ÖËr@$^2$G¿¸**)ãè'?û[   ↩
    ÙÚIêá5+#a$^3$&p?éìiÉ}«·?0?
?!?áà(¿í
Qú®Åxù{sç?fï??uaNJ~MY\?x|j~øTAW ;>>Iº$#ÚÈ'_Õê?#?9?}wßè?:e*#?ç¨gJ?|?b???OíÞ$^3$é? ↩
    ww¡?y?$Òj"RÐ¡£H~0éORT?ÜX-ÙlË~Â?":î?p»^?i[Î',ì?3@ïVDÃ?7£?BfÔH??$TpäJ¤)ô??ö '? ↩
    TØXH
?ÀÊÚiÜq_ëdQû??%<"Gx$^1$3(¡/\®ó¤¿»?z)y$èÉ\textdegree{}IG.4uº ?ª ?)$^3$v?§Á?Á*p?,| ↩
    ½æ;Äßä;&]ÚáÃ¶[c\textdegree{}Q«açwXë?8X;ÒãhE?Mû9'KÉ 47Í?>FpÒ@jl-8U)\ensuremath{\ ↩
    lnot}íÔ??EÂ©N çÍ?ð .$\times$¯ý0ÇqÇ$#j?HÍM%.w«$\mathrm{\mu}$.)d?XÅ8´½Ç?w_Bs$\ ↩
    mathrm{\mu}$H?æú3?Ù$\yen$®?Ü5Aû¼'øÖ´ü;Ê#éK;h:NÃ?\Øº´Y1'$\mathrm{\mu}$t5í?E'$\ ↩
    div$?0|u¶
ÞÚ$^3$,ÞîüJS.&~K;?RdQY%¡$^3$$^3$T?e?wU$\mathrm{\mu}$?5PÀ??Û\ensuremath{\pm}:oÞÆ$\ ↩
    times$Öißókeáèmi$^1$^j?$^3$Û£·@\ensuremath{\lnot}1???&?©¨C?ÞQöãÁ
v4$\mathrm{\mu}$7%®¡$^3$D¾Ä «
1k98·øa?;?g?$^2$\ensuremath{\pm}IÓÿÛä?fÀs$^3$\ensuremath{\pm}?¸2Ì?L?bÈ{ îÊY[V#.?Jþ ↩
    +.U·Â%Í$\times$ÇÒ?Ì>/Ñi/Zê¶NäW?}õúÐ1<õ$\times$àafgè7tý£??aï?y?iÖÓÓgb?(\?ÄÐ$2"4 ↩
    $TdNU_?çÛ\ensuremath{\lnot}?1+?¤N? ?m
ëæ?¼F¯ïæ\ensuremath{\pm}ò~Ø.Ûdl@Ø ÿF!'??<tüá@QÎÐP?4ëYËøoÎÈÖ'??QPÒS?mM7+~?$\mathrm ↩
    {\mu}$kÃãWP?C$?Ìý+$^2$¼nß-ÃF?´t¤oB65'¼y,WÎÝêÄ! Ê3q ?PP?3EN Z[@íÔWÖÙY?$\yen$] * ↩
    j9ú?@oïKÕy¿¶ãS9{wÐ!mf4ÔÑÄ?c·M?*Rn‘ã%7%EßZ??Åó$@#h  ´1?¨«>5Ö£©¸$^2$ä)~®WcHÉ?ñ[? ↩
    ëh¯ÂÌnÓÓ?F&ÓwÏYËê#Íý]H\ensuremath{\pm}M??97Ï?;iè4?k»F«?áÜ  QsÃâ+»çòz?Ö)?ÐVÑHrÖ; ↩
    ÕÛ?üëVäd=¾cÝi¨áã$\div$®?!<dìH3CÌG!
  ?s?¦Îo¶ÞCeû¨dÁ-\textdegree{}BÐcÕ©ökìÎ$\times$qÃâü??U7\Ïà¸Ô?C?ÍèÎÏËÖcÇ!´Vû?Ê’&CáK ↩
      <~éº
?b??Ið??2‘+?ö·Va?A+TTðÕ?t??"mE}¿O
ù6SWÝÅy$|Ñð’ÑÖ>x$\yen$?¢Dç
??ùÙ;Æ$\div$Ç(UÛég?8??ò¤É?ÎRTÕ?ìé?&O!#a»’,$^3$Ù@zÝÛòc!?ÉÃ<hy?§?78nYQ‘?I?Ì¨??¿pÖ0$ ↩
    ^3$&Î?HYäÁR¡Bk*<4’çª.‘Ûívºy
s$^2$Zj;ýxPLJQ> ëÔ&~=FxÉ´3$\yen$7?’?YÆf|OGKW#QÑ©Ò???p*ïd*Ðìî???8?çéJ’=-ã]Hz¸K´$^2 ↩
    $$\times$fgkÉØ{$^3$$^3$+jTÑ ?*xÉ¢Fn<©Ù©q!ëhÃy!Öä?ÅÍU ã44Õ(Ï?tÈèÒP??Â??BzÁRìÌÏ&~ ↩
    tV?ÌL{,$^1$ÃG’ê¸@Ð¿©ã?¡AVpÙéøð\¸fÿ%wûpe?Êj‘i}SÙ¡$\yen$6xí)ñx?$\div$3Ø½é(ð~Ã???É ↩
     3BdSUH?Ü\ensuremath{\lnot}ÌÙAÃÈ?c|é?úr4~<?\Ù3??’F7.
Ø4ÕÔ  PÐ?º.îüÉ?ð½?|\ensuremath{\lnot} ?Ç¡?/6z$éÎ?Ê3©§JÎ?vãÕ}©ücbãdâ)gB¢$\yen$¤$\ ↩
    yen$
J«?Ó& ?$^2$e$\yen$q?ì?<kÿ?}k?8ÄÚzJx?ågÍ*º¦ÉÁÕßd?Ê$^1$ªÏ37?;)ÎñÙÙùÉÊ
?ÕOÐÜÆ$\div$ùÁÛ??MÌRc$\yen$Ü$\times$Íñ}$\times$«t3t‘Û~è$^1$DEÀ¤?@ê$\yen$
Yp
Ï @éñfDÔÓÉÕÌÊ]?Ì=$^2$À&GJª¶ÛiYoclz}??GO$\yen$8_OÁüß\textdegree{}Îg¨ú<M7$cÎ?>¤ùÓ ↩
    #4?2Í??~t?ä£:\ensuremath{\lnot}ÕîÔÝ(òæÎS$\times$Q\$\times$$\times$ZÚV¦?%$HMBÙÕ· ↩
    ???O$^1$úy$^2$5Ú??Æ|TôTõþÕ%j$tw!Sëø¤å
?ã;ßÏ?v$?-ÈÎÇÛ/_obsS\?â[*ôfåjìÁ4
Z
ÛÞ¯û]ZALI;:?\ensuremath{\pm}Wt?ñR$^2$¦®’??¤TGÂ??Ñr?$^2$XgJÈz¼»fÌF¤\ensuremath{\pm} ↩
    áp¸$^3$$^2$?2Å¨R.+¶+ÐC?lk?_úäH|Á¶»ÝàDç©55Z?Êª BH)Qò??^l?d$\div$o\ensuremath{\pm ↩
    }60¶á½‘BTH©xuá{?çÁ$\yen$>$\yen$0?ÌO?à#
Ñ6ïcìÂ?Z??P¦·$O‘}üÀ?
,¸À%?ã??KaùÀ.z#[Ö86?2fîn0\ensuremath{\lnot}¸ÈÐ’\ensuremath{\pm}T T#§KEAÃ´?´?ßoMÍ<ä ↩
    ?<6?L<®º#’ë??+=>
?âÀâASÅJ¾È|C»à[$^1$’GbI)3¡ûZ0jìÃ§
???:r=u¸9ýoÆZW¸ës$\yen$$Q:vNó]n?@ÄÇÕß>læì?ûÅ  Í?|ß‘HRý??Î#?Ùà¸Ë?r"
½0TF?Ñ%<F?ofeïS¾@ë$\mathrm{\mu}$ØXë???DÙ$^1$1MhË1Ô?ÁÕ??dd‘?Y:?ñºØkcËêhJ_??5üÄ©?|8 ↩
    xÚGo?VÃ$\div$aO8Ùn$^2$?Á<Aöî7A?õ»&MP?,½9?
%8m§¤?T??ózÙÎÎIaI/?$\times$v¤êk-%?$^1$#G!Y^w$^1$ÕZdöû?«Ïò)wÀE9\ensuremath{\pm}¶60  ↩
    ¸de@Sã?AräÉB$\times$g»½1¨êð$\div$0û:ø{?Ûº:!? Þ_?¡W?tÀ?9uxWõê9;‘b?9pé©*ÒKóe>? ↩
    ¾MÃÓ´???læ?Z®3[lë&Lí?»/YgÀCN+«»ÎW$\mathrm{\mu}$ah?ñ\textdegree{}LË,r\â^$oË©? ↩
    ÑXjINlõ=Iu\ensuremath{\lnot}wÂ¨?Õ®á¢b~I?SHRHÈhÙÖÓ§Ãgò_Bì$\times$ºÒ¡ï
Ãßÿ?|:ÊðP|j©en·õ?¤?fOâ4??¾7é$\yen$?XÞìÈLÌ¸Ì4&OníèHpö!óòá?
Å?~7!?  ÕBt}ÐíÓyÚzxûÔá«?$^3$Õ½k:uè$\times$k?é3nÊØ)gÁëX]éÔÆõÙë??B½Övüò.?C
9!$\times$í?Lö»Ê?F)?bÎ(@ @ºÀûdÞï¶Ó
ÅACÜ?F¸?PÙÆchdÚÚÉÇÓ?ýbï$\mathrm{\mu}$É?á©ÜìXË$\mathrm{\mu}$gÌ?·á¿ÆÔV??K^?Åôñ?>?? ↩
    JtY=½ý¶Æ{XørÖP~Ä??6·À?GF?ÞìÙXq~K3?ûÇ».#?d?£HrüÌQgÏC§9QuÊWaÍ¶©º
n~;H¿èÚ´??fÏ?«Dp$?<NÌ?\ensuremath{\pm}%ë? ?<ÆÃÌx<$^2$]ë¾w¸)]$\yen$ÚzåÌÏÏ?ÂK??üÑ? ↩
    Á¶¶?Åä?%XsT‘þòÄÉ4Ù,N}?
:Qt\ensuremath{\pm}$\mathrm{\mu}$ÍÕråþ¦¾¼vº?ðÎÈÝ$9ÌYhBÉmªõÉx1$\div$TX4öN? ↩
    ¨EP«sîÿa
»ÍÚÀ$^2$¿)T¤K??ô(¸~¸c¦îHè?ÙgùÚ?wêÖÜ
$(Nö?ïÈtÕîÈ$\yen$åî¿2?bZRç´4¤$tmø·
ÊP«?â£aÖe?{Ce¦nP©åp:ºÏzFÓ
[´ÚcàfÿgcVKf??5?É~é?\ensuremath{\pm}àÈß}?*Yp?P*?aåJìøÿB´~\ensuremath{\lnot}Yk£¯¡ # ↩
    ÓRiMÄQÎS6wgÕ??8[-<Þ^6?Ö$^1$Ë$\times$¶$\times$qªÚ??mâo?ß‘  ë?  ò???Ê¤?í:U|þ(NG? ↩
    eËþÓ·ÍÎ?;???«HF[M2Ø?Ï$\yen${}ÍM4u¢ÊDVí¿Ö}$\mathrm{\mu}$?|$^3$’e6Ì?z=Àud?5?AÇ«? ↩
    NE¯?ã?ÄÊýjãG~Ð~\ensuremath{\lnot}P?¡~?~¶Â{=þÃ]MF?Kr?fèÆ§½YàÅ{»’gp\textdegree{}ò ↩
    %O?K´$^3$h&?H¦ME?Äït?|OzËÑÏÂ&ªH»àÒr¸û$^1$Ð$\div$??vFGj3ý?/Ç?G7*??Eõÿä]Ç?¢î»j ↩
    (?¶&?5§2Ê´tÇ6¡PÂ$\yen$Yô?[ËÅ?‘?@?$\div$@?Eªù!??Óè\ensuremath{\pm}^8¿À{§ýby3ú\ ↩
    ensuremath{\pm}´ÌZ» ¦Ãýy%H$^1$Á(Nr%?´âäØ?1Z>Å??Ç?æÏæ?$^1$#\ensuremath{\pm}ô@? ↩
    ñÉÊ?ÊQT?Íæô]Ð?1øW»qÑjÜ\ensuremath{\lnot}ÖM?-ÆÌÆdÖ,h??Ù1ÛL\ensuremath{\pm}p’ÓÛt* ↩
    ¨ãÖ?¤?®(}? Ç_X¦N^$^1$1??4 ùf?¸?vÃ£8âú$\div$?ÞZÜ/?£Ïà]¡åñßx:úK7G? ↩
    ¡4wª¶ÛãFhIfL§®©öù$\div$Hbm¸Ó©«,?Ù?Â
{S?iâ,?"öÌó\textdegree{}¯\ensuremath{\pm}?ùpõÞbã_ùY½?6ÛFQàdÈ>¦\textdegree{}*aÎF[,é ↩
    ;*º{«bË¾KóªÔ?]Ð£Jhã¦0áêÅ?_åÚ9??¾u«Ù.a¤·??3%Ð¯^íîÀ$\ensuremath{\pm}?é~¡QïØÏ4·?ÜK ↩
    ~YðB$^2$©X$^3$?~¢|??
?yå6$\mathrm{\mu}$© k??bP&4rñ2;«$\times$z|w$\div$Þ,L,?/§·¦ýÚöC7?D(  ?6?IÂøDFõ?FÁ\ ↩
    textdegree{}$\div$R!$\mathrm{\mu}$é
?S3¢r*©ç?Æ§®¦?Ö?ÈÀ$?'þw\textdegree{}jÃý½àýºëüoV?
Îäp7? 9óÆi?¢§åªUôÖÔÅ
)fN?$\yen$pÜ@9Pæê,¸g?DJmZI|ú?ú?Pp¼?ãúp1?ÞÖZü$\mathrm{\mu}$¿ËW:LÚ´tT¢äA?x:õ!? ↩
    @9YUwÍºX$^3$çk???;?ää$^2$aÛÓy¨?S6t;]ì¯?ö?¾ï_?Óå^$^1$|
¨Zº?£?lÑrÍEtj#{«¸VA?2wnÿô]îo?_? ??+?ëUCÎõM  _8ûÑÝîd4õæhyúßo¿·ùêßâf$\mathrm{\mu}$2; ↩
    á??iévºªØ-ð$^1$¶$^3$ðÍä«\textdegree{}?doU1?Èá®ã*9eÍ¨?eºéS´'¢dáãäßbx»Ø_c»Þþ| ↩
    ß½îSm33w,
óê)J*ú*?]pÊÂÜ?B??~= çJã/?g3b%-?"|?ÐëÃ?aÎ~/vû+3??ÉÚýº½õïôÍw®ýjÝ?ÀJx^nY $^3$jj?
$^3$Ûº7P?´©}ëÅk.ê$\mathrm{\mu}$Ü·ù?ÿ~7;F0?3oä¨?aj_-ñâ-ÄÌ[ÝÀïÿ¿¯ù¾ÈlË'ÇÑ¾Ù¡åI¸=0 ↩
    vT¦P?võÊ?<?1gÓx_$\yen$8sáús®?®Wð+ãÕÇ4Jx«\ensuremath{\pm}òqf3[á¿ïv¿>Ýï3$\times$} ↩
    $\mathrm{\mu}$}óMà?ßÙ2|?¨O£:?Á6VÔSÔ\ensuremath{\lnot}$^2$?Ô}ä£*$\yen$eÉ¤{Å~$\ ↩
    div$6¾??ªä:[üøs3ÆýûÎ·å$\div$ÞàùúßV?^
ì|\ensuremath{\lnot}Ò¦¦M)A~Xø$áã,ã?ÚJ?$\times$øÿ"5m?½àÜ<"v*ûçç o?MX"´õvüz?3 ↩
    Çþþ¾§WúÑëý»?v?/ëÒ4ù{eæ?¤«$^1$¨?ðú
[\ensuremath{\pm}~Wuqòì1?ÏDmj=$\yen$!Üü~ý}
=TTGÀëófzÚ¸Ú?i7ì¿«Þ¯üû¼?¯?ñêÏÒfTI(¨U%´?ÿ$^2$?V$\div$>*',øÀ¨U?\ensuremath{\pm}9\Õ- ↩
    «®ÎùîiM¡T*Ñ[·òâè7¶ÃôêtÜ+ïþÎÝï_©$\div$'bàøü¦OeI{´|?*¤é?;kl'»yyÊIN¡6¤t?6c/¤K;Î8 ↩
    >???GÝ:$^1$(º¾,5Ù:Ðø>Þé|ÃÏûyá??$^1$L$^3$oýo» x?Ð£??JB$\yen$8_?Ëp?$\yen$ó* ↩
    ûïë¿_ùYóÝðÃ?e8þÁÍkþêíI?$^3$ºýÌNÿg?ÿ¿¿ÕìøÛ$\div$ÞKülv
3Ý?(3~??
¾}?ÐTb%âËÊUs?|?
Ãë¨y?ÞzÅq)@Ú??ì}îï_Òü$^3$ïUï§?ñúüÎïËöþ$\times$ÝÎ$\div${Å¿?$\div$?f8§?£<0IíT¦?:~ ↩
    gÃÐYÉ´po9ý(}/¤%*ªÅ:Ïô!Dâ)S@9ÿ}À??ù´Öþ?_õäþ?$\div$:½?»ïë__ââ_vû¢Í?Ù?©®ÓO?hÞXØ?Ú? ↩
    ÃòV?(Íf¦? ïùYõW?2
8V(Íz¯»?Ù?Ùè9aúv/|zN0¿>ÿñØðgåä4Çív0A$\  £D&´QÁ2¤*?WC3e4"?¤???Üá(§j¾!¿ÅYèc<
2?þ»î¾C½OÛ$\div$úÿs?·¾.$\div$ï$\div$ùØi/òö¿^úßìOä#ÈÝ
ÑPI¨SØÖW{~ïB®?%4¶ÞòÅôHÒVrï?Ç?|$^1$?4 *NQßÃ?v6$^1$7Þ¿Zÿ?7?Ãàü?ß$\times${»/ÃÞÁÇÁb) ↩
    TÒÇM5Å?G}õtkËË1$^3$îA(b{ë\?\OugÔ¯6«v?!ZOí\ensuremath{\pm}ÈÀ\¸~óûÙ½ý|?$\times$$ ↩
    ^1$â½êtüØ?K|ÝÌ|E$^1$NÎ???ÁÊJ»kon^??£ï|)?$^2$'BïêT
¸>o?Û¶öì*?$\div$Ó¢SYºÖf?ft6»-$\div$û¾¦?7ý¯Ëñêbböï¼Wß?[º»0Í$\yen$?Ò¼\ensuremath{\ ↩
    lnot}í]?Oñ^ze?ø\textdegree{}?âúî{U*.?uSIGYñóhÒ?Eë¦k_G ç?úûÁ»???ö?§$\times$·Þ\? ↩
    ÎÆòæ$^1$ÒÏk?#É~î??S<¯ÐYñª?xØùéCIÌ?Ç\' $ \mathrm{\mu}$UEQ?èzlÃß\ensuremath{\pm} ↩
    Üîø4?ê®Áíuz&6veÿb$\div$?k$^1$béëëkn"£ª?$\mathrm{\mu}$?ð77V?$??A¨$^2$\ensuremath ↩
    {\pm}Bjí>Ð\ensuremath{\lnot}#+´å|<#v$^2$WêÖ$\yen$Øê
§ç?Ö3èº¾&?5þ?û?f$x[Ú2%?ªUL?E2{$\mathrm{\mu}$ª&»â+Yòñ>??61´þ?¡]
n%=(ø©b9d·N&óNç¯$\div$_óâ<Äýú¿]íî?PÎ¾½ÒâKíGÊ,?"Taðý?$''Q?¯ßÿ??Å?¶%Ò4¶i¸|;ÀÃ$^3$?? ↩
    ý\?>dX?´Pnâ~}ÏãÉçÂÂ]áü?!üöÿ+r0/¿ë?æ?È67ÀÒ
ëô}^íØ&$\div$ª,¿ÈØ®N6ò¨¸öÓÝYWpÃ$\div$?¦?9)îÔ???øo{ýß
æeçíû?ôgÈÇÉæÉ[êôcêB??$^2$s85Ïf!¢??x6G¾p¸vàò??B» ??ÉLàJÊaÁ£Õ½o¯ðíßd3ÉîþÿÞ«?¸C;W')c
8¤wK?Rjp."Å t'
¤ø
?5\ensuremath{\lnot}¦?Kz¡WLt++:??¤o$^2$Ñ$\times$N=7¾/7?$^3$L¯5ïÿºßÍö?F~f;6 ↩
    ëÙ¼pæ$¢¦àÎ«¼ú>q??vÓß?^ßN- m]Ï?A9úÑ?Ii]UqPx?R'g1k§§ãiv/»?L<L.ßíþÆ{ßk?5¶&s?¦¶¡îÈ ↩
    <3Ðgñý=?M $\times$e?$\div$?8;?
TN.J«)Á$^1$ ~
¾ú]ÍO6Õoçìâx»>kìLNÿëötþ»Þî?$\yen$vnGn$^3$V;QäG?LüýO?¡ôs???%ß·ó\ensuremath{\lnot}ù ↩
    ~;?N?]Ç"ïwebÌ0«ë??þëfkó?z<¾/?[öÁ[äðx?Ù?¯üàã6u·1{'ocÉ
*ûpúWõÊn%~ÝáyðdM
}$\yen$=Ô?J??n  ¤e\é2k$\mathrm{\mu}$?!$^3$E½\ensuremath{\lnot}u¾¼?¾/?wõúû·ÝõY[ ↩
    çò^í¶B'i+Ur9<J?G+ííy_7$\div$¾xÙ$\times$Îp¾$\yen$a®º~Ì5??s¦©hMë
.¯S?N’«?ñþo»ÏÇ{âüo|?Ö?ÜýwNÛ$^3$ýÜ?#Í?iOgîÝ§%9$!.nº?OGÜ¼øgî(§$\div$ùün=?o´ U©(H#´? ↩
    âëõ$^1$~¦9¾7{»{æï_uðÿ$\div$§?$^1$ÍsgLX/dßIãdÒ)¸\ensuremath{\pm}$^3$¡1??¯wæYyî,¼ ↩
    ??ôý¯Çð]ý6t4ÕTª))?0?ñÿù~?g$\yen$nSE·$\times$ß§_$\mathrm{\mu}${Úþ?Ï½êðx½?\ ↩
    ensuremath{\lnot}=’??6k§»?q?\ensuremath{\pm}$^1$öÂ?¾
Eï$\yen$gÏõ|·?[p½]?q<ÝmÉÁØ?
??\¸_Ã?æùx]ï?g½øy?þ¿óÇýy0W0ÁÇo$^1${W?NÏWÛZ??RRÚòÄYÿÿï£ÊPÈ¶âô$^1$vaS{%È??ù2É ¸e·=o? ↩
    ½{$\times$íw0ï\textdegree{}¼~U·ì3ó<y¦ãU¤ID?ÍB%??Ñ$\div$ï?öÿKñT¦Ur¾Xv?´TIÐ?¢?í ↩
    ???? |Í~CÆ^L|??õÜ½ýïü~$\yen$¾NòßG¡$^3$=! Ú):D~$\times$5
DÙz¯óÞ¼ù¯?ÜÇÌ$^3$
=?Cà?Ea?;Gì6?ùùk¼ùY$\times$Ìù¯|W¿Ï»Úÿ??,¼\ensuremath{\lnot}$\mathrm{\mu} ↩
    $ËVãi6Ô¼ÝóÉÔªK8R@\textdegree{}Á¸ùî$^1$?Î¼$^3$êøöíÂB¢ÂmgRÕ?Cì#6|
¼æ’¢6põ¾~k&x~?îßwÉãÂðú¿ûÑãÃñù$\times$6g\textdegree{}ú?-½ê*9É"¾@5+g8oHý?>=¨uT$\ ↩
    mathrm{\mu}$’AEK‘?ùï??uTN[A]}$\mathrm{\mu}$¯úîà$\div$\textdegree{}ò3Zäú¯\ ↩
    ensuremath{\pm}bº?Y"Â$?%IOt’KÊå«öù$\times$/ÃÆ$\mathrm{\mu}$\ensuremath{\lnot}B ↩
    +?3t’PU¦¦J_hWP´ußHÝó,½?ËZhzüu¸^mVyl?wü?¦
?8
ì- W\textdegree{}õGp:+.ÕÎø^rRÏóP?m)á?UJxábQ$^2$W$^3$àç¯3$\times$ëò- ↩
    bÍrÜ_BÕíYikkz½Ks?AÚnt|}È[P&?4u9º{Á=¡9sÕôV|\!?PÊ? @¦lñâ?ö?\ensuremath{\lnot} ↩
    Ù©0a7_\ensuremath{\lnot}Ã=$^3$®$\div$???!¦hÞ¡¾ó/ÉÑ?t$^1$LY5$\mathrm{\mu}$"Ø#  ? ↩
     \textdegree{}$\div$¯:
øüÏî5?½Å$\mathrm{\mu}$
xøÍ|?ÉYÐ@%·ê+$\div$Ð?\ensuremath{\pm}ôée.ØÉ¾^[Fî\äáãf¸ÕØß(HG8ÚÚ¤fÔ?L@ÕÊlxêê\ ↩
    ensuremath{\lnot}?åqÅWqT¡@Í%:???d?\ensuremath{\pm}\ensuremath{\pm}ZÕÀâbÊûYÎ¤)R\ ↩
    ¶j6Ð$^1$í?ãä4‘Ñ8eu?åö=?\k?J?ôG¨F®ß=_?îyñü?$\times$.æÍ8T´ô?Go&Û?/W<nóñ¾v¼]]Ð< ↩
    ØlçaþÛ¦ºc|‘ñfãS=¶¼2#ì¶? ¤áô$\div$Uub]«åò¯:7ckáqiqc\piÙ¤?Ù$\times$Öo¡?.ä·$^1$º? ↩
    ÝjE|ÛC?3?v[lìý?ùl\;nÛK%\ensuremath{\lnot}=wçÐ ?t?VØ[T^þ/5_?\ensuremath{\lnot}:ú ↩
    ?RuI§¤,XÉXÚkó\textdegree{}õ´´Ç9»?ÏÛ<uçZçQ$^3$éQ?J??xïcl>Ü<É??á¿)ºu?úoeE?M:¡ ↩
    -?????&~???H????????ô\ensuremath{\pm}>??&ê?&é?&è?&ç?&æ?&å?&ä?&ã?&â?&á?&à?&ß?&Þ ↩
    ?&Ý?&Ü?&Û?&Ú?&Ù?&Ø?&$\times$?&Ö?&Õ?&Ô?&Ó?&Ò?&Ñ?&Ð?&Ï?&Î?&Í?&Ì?&Ë?&Ê?&É?&È?&Ç?&Æ ↩
    ?&Å?&Ä?&Ã?&Â?&Á?&À?&¿?&¾?&½?&¼?&»?&º?&$^1$?&¸?&·?&¶?&$\mathrm{\mu}$?&´?&$^3$?&$ ↩
    ^2$?&\ensuremath{\pm}?&\textdegree{}?&¯?&®?& ?&\ensuremath{\lnot}?&«?&ª?&©?&¨?& ↩
    §?&¦?&$\yen$?&¤?&£ ↩
    ????????????????????????????????????????????????????????????????????????????????
    W?&¡ÿÿÿý?????&¡???H????????ô\ensuremath{\pm})ý?&2?&1?&0?&/?&.?&-?&,?&+?&*?&) ↩
    ?&(?&’?&&?&%?&$?&#?&"?&!?&  ↩
    ?&??&??&??&??&??&??&??&??&??&??&??&??&??&??&??&??&??&??&
?&??&??&
?&  ?&??&??&??&??&??&??&??&??&??&ÿ?&þ?&ý?&ü?&û?&ú?&ù?&ø?&$\div$?&ö?&õ?&ô?&ó?&ò?&ñ ↩
    ?&ð?&ï?&î?&í?&ì?&ë ↩
    ????????????????????????????????????????????????????????????????????????????????
    W?&¢ÿÿÿý?????&¢???H????????ô\ensuremath{\pm}?ã?&z?&y?&x?&w?&v?&u?&t?&s?&r?&q?&p ↩
    ?&o?&n?&m?&l?&k?&j?&i?&h?&g?&f?&e?&d?&c?&b?&a?&‘?&_?&^?&]?&\?&[?&Z?&Y?&X?&W?&V ↩
    ?&U?&T?&S?&R?&Q?&P?&O?&N?&M?&L?&K?&J?&I?&H?&G?&F?&E?&D?&C?&B?&A?&@ ↩
    ?&??&>?&=?&<?&;?&:?&9?&8?&7?&6?&5?&4?&3?????????????????????????????????????????
    W?&{ÿÿÿý+-,é?YÚÛ?4?Ëî?N?Ð$\div$kóván’\ensuremath{\pm}«\ensuremath{\pm}\ ↩
    ensuremath{\pm}$\mathrm{\mu}$?,:\ensuremath{\lnot}?4jÒ?y?43öRöm?5{\ensuremath{\ ↩
    lnot}ßaüá\textdegree{}&¡¦£\ensuremath{\lnot}?N?¶?/‘’"Ì>G 1ÃÅdA?".öæÆ{??öOåÇ?) ↩
    Óâ2?&BpW/BD¸?$\mathrm{\mu}$^$\mathrm{\mu}$Ó}%Ö¤?ü½-b?#?|¦ñj??*¢£?z4?ð Á?¦}?ñÔDã ↩
    ?àÜ?_]îÃù$d@’/th$\yen$I ´"8î#ÌÂ?$\times$:õ!nNÀàI?}%}B~?
¡E]$\yen$z~B¤?$^1$c?\textdegree{}?ì<õÛ$\div$û?E ¢?Á\ensuremath{\pm}ØXñÔT&Í£?tÅÈ?,G ↩
    ;.?Æ?X£U:dâCã$\yen$*Dú4TT§A¡<ùÓÒggI*aÐÀ$??D ???À?‘? ?4TY§Z¯Éo:y\ensuremath{\pm} ↩
    ¦·F)MÈ!»?ñ?ÓRQ¢Þ1"\åE0J?+0áã#)B?ÑÞi7w?Ã’N9:\d?%
âTÉh\textdegree{}KoméH\ensuremath{\lnot}$^3$ôômL}?D?nÊÜxá?Ó\ensuremath{\pm}sfI?$îô ↩
    =ÂF??!’C@ä=Q¶ä¡
Âá0Ô$^2$GIt©?~æÉ!Bp$\div$?Þ?g$^2$5a
[?®4ÉÚh?%7ÐX*§&ÿt#?¦Ñ
```

```
Ú%ÈCc$é¸N?f\textdegree{}};¨Þ?áûÉ½?SÀbÇ_Op?:ÎbG!mêgél6Û9ºí9ª2\textdegree{} lÝ?(Ò ↩
    (???ö!?£{?NÔ (Á4ãtmu]¸}??? MH?t$\mathrm{\mu}$õ¶ªÕO!C‘¢îwþÁl4?ôK#ècë%BæaògNMì?M= ↩
    @XS¿Úw?lÍ8Û-¸Â^;·MÜD4?#E?%?zë:«a?\textdegree{}öjÄQy)ØàZÊ?-Â8?Z·?1¦I"n¶ë ?ws5x ↩
    ??8&üRÙ®w©?+$^1$??Ç?^ð?OTPS&:¤fPVq-ìU?4Þ7öÉxs?f½©âÔ?Â$^3$)V]??ogÄ fI=y"ilK, ↩
    ÿÿe@£?s·;Ú:??Ü¡6]J?Ê+TQûÝ?ÖYr$^1$ã*Y{ÕîI$^2$º$âêQ$\div$Ê9ZY\ensuremath{\lnot}õ? ↩
    xé? 4ÐÛ3ºkÒ(ê2Ó?ÌA?|ÉrõG?~à?¦Fr4´aüN2{Q=Ê?É?\ensuremath{\lnot}g?{mªÖÉ)¦?ÀußÅ7Q ↩
      ¤H7?q¢à©ïÝ?6ã^<?û1ö‘9q§=6ô
??-8Çe¶ø9|;?óxpHî´Ç^???a9õÆ?Eä~Âa¢ýë=Ç??¢IÏË  ûó$\yen$?K?É©??MèF\ensuremath{\pm}> ↩
    rÚ7H®$\mathrm{\mu}$O$^2$?~Ð þFD&$^1$ÎWº O}n+*q?bA  ¾*x(0~ÄÞ
öc>Ït«·h??/<© ?·4Y?C@X?e"UP¦§$^1$?ÜÏdBÔ7L#¡©uéÕQ’_v?PöÛ$^3$ËÞ2eÎ]W?¾7Quv¢3Z(???@) ↩
    ú§?ÕOïWZ?(Ï?H£à
PìÃár¾>#??æÌçé_aèW@pì|É&=Ô?ääý=í\ensuremath{\pm}?#&?ÍºÏ$\times$p,C?n?2?YeJ?@n®¾ 5¶ ↩
    ?W>ß9gÃsÏ¼(úP"Oâ0M??qw·%fAËËÖ?6O$^3$òÛ3íú$\mathrm{\mu}$¸mæßêk"TÄe$¯~xÄì*#$^3 ↩
    $dÒ$\times$ÖÕ??UÖw?âr?VKIÊVÑ¡ÃÀÎGLýÓ©â$ãìÏ\textdegree{}¿Q?:< [V?Wéè@]éÂ2Ó??Ô! ↩
    Ïóa¨¤?¸A??äç§8ü?Å?N(A\ensuremath{\lnot}á?^\textdegree{}HóªìÓg9·
Ä?ï|ãtc3/ï6Åîä.òå/ÃÆoånë36?ü}ørÒVÑ$^2$?e"sRRÔRU©??@èªùVß0Rmº????k!îÊE??B?{r>\ ↩
    ensuremath{\lnot}MMgæõðòÛ;ñéyð$\times$\ensuremath{\pm}Å^Û???ÎO.ëSGË$nÁç?B®
T?ÎgSJ?"ígúÓ$\mathrm{\mu}$å,,Z¿Þá?j«?ÑàvâÎ7+?£,1ãv2ý]$\mathrm{\mu}$¸ýÿ?F?}nî?vÜH: ↩
    ÎeÓQÆCaCV?ºÂ<píAD?Ï©á\ensuremath{\lnot}uJÀ $\yen$?ÉÂÄ·K.¡?/Ë$\mathrm{\mu}$?? ↩
    «3þs5<7Úæ?¶?ÏG{!64uN?Ò&Läà -!1 ¦@.jÊý-V\textdegree{}úEß?Ø)h$^2$|?´?ÐîæLÍìÌ_ÿvük ↩
    ?Ë$^1$Þëà~½E«ü7$^1$/?¿l¾M?Æ´£>Ù ar8äøÀOZ87?gÊìUK-¸?ÓóVp·?ù}ÓE’¤Ø$4£’@fÇ®û! ↩
    Çëøz0/r¿¯¿õ$\times$ý44$\times$Îb:T8ÇID4ý?Vô  Ì¦öé??Â»Xue?Ú«?§à¯?¡YÅ¶§¤D¢)o?ÊÅm$ ↩
    \div$nÿÉ}öx//?î1??x;88YÛçG}’p?vçÂL¡%o?ïQ©¦?Ë?WIÚeûghí=ÔÑé?X$\div$o?¤åS?bZY?X¸
1Vê5ûþÅ¾·Z$\div$ìû=?½íþ?Õ\ensuremath{\lnot}\textdegree{}ÖH$^3$?B
?Ë+;;$^3$öÅ=õ\ß[ào8dæZ\ensuremath{\lnot}$\div$?¼X’ËÉãZØt\ensuremath{\lnot} È <®êï? ↩
    ïÙùø¸è$\times$Á¾ëv;ßÕû\ensuremath{\lnot}æ?%nPô5*ÒØª ¸?Ob4$y$\div$?b{5¿!?Òj?t?% ↩
    _Ð$^1$èt??A*ùU&£~·Ó¤ïÁÕìø\textdegree{}ÿ?¯áõø;\?¿ãÛaæ‘Ï_^C?æÌÉcÖa?¯æ[QWÐ|
þg
?,\ensuremath{\pm}J?ÕÉIAyÎ´¼ú$^1$UWw?xq??¯-kß$^3$$\mathrm{\mu}$ÖéýÝ·_$\mathrm{\mu} ↩
    $üw?_Ç?$^2$ÛühÐ?Ë m?T4¡SZYP?àVte?¤ùl??Ýàãü5?EXZ+?Õâ$\yen$áËÌvëYïÊ$\div$
$\div$$\mathrm{\mu}$öþ]^¿gò½ûºÝÿG£ÃË^?ËÁ?$üõU?<[]XûÝ?ý#«?~~Æñ*?Ö]Ûý?OÃÆâ«Yó?.H?? ↩
    ítuð/o{¿‘ÞéôûgW$\div$ûºkWv½
tòß9 ÓÞ?JC$^3$æîW{Vüè\ensuremath{\pm}èî§g%D  ]#ÃêÕþxõjï8½%?ë\ensuremath{\lnot}ªÊ> ↩
    Ô$\times$ç?Ãßêßø:$\times$½$\div$ú_oüÁÂìâ5Ç¾_´ûFn?Ä(G¤?í
}x¼^G?«?tüºK«"tb+$^1$¼çs\ensuremath{\lnot}º
ø7-??º¨ó\textdegree{}[kKÇíáå¿Ëïûþ¾ÿûéþ?.·bû¯àÉ_$\mathrm{\mu}$«’F@?3ÓªíªÎm-9^?5EÍ ↩
    ?d?)O"ª
%º$\mathrm{\mu}$u+O¿Ë?:ºÊ?E·5ß8càïöß?oîéÿ=çc¯Ûô-ðã:bÖdhdÍ?By?Õz©?Kr¨?ê\textdegree ↩
    {}éQP?$\div$Úkj;Ok¯pªá_>æÀ@¨jh$\div$£Êàe®bøÿ?¯\ensuremath{\pm}áë}$\div$½Ë\ ↩
    ensuremath{\pm}ù?X?Vl;ÜrhT5hÍ{5?õu[~’
+?Ð¸T)=?R%MhbrílBçó-Ä$^3$öÂ¨¡?láÉp5êbàø2w¿?ß/r$\div$ïëx¼ýßÝo??{-8\textdegree{}Ê+4 ↩
    @å
E8VÜNE½$\mathrm{\mu}$Ï K«\textdegree{}Öü
ÕÛqD$\div$=ÄõªúAð\ensuremath{\lnot}?TD® É.aºÇcú^ø/úÝn¯Oð?½ß·ý$\div$|?}:?NaL0@Ý??U ↩
    \\ensuremath{\pm}?ò,ø^-uß?d?jk?5~Í?ÎE£KÂ??ÁX~<pÀ¼??W?\ensuremath{\lnot}Ùâb?$\ ↩
    times$gø?¢’??¸‘T½ï@~???ïü??w???ï9ET{ÎeUV®ü?NkQðU o??ÞÞÛ{cäö$\mathrm{\mu} ↩
    $¯Ü¶ÛmªȘ$\div$ûMjÕ^ü$^3$U3;«+*þ?UTSÂ????t$\div$??ÃK"=?û·?½ÇO$^3$Úòdz0n$\mathrm ↩
    {\mu}$$\?óé«D[l?ªþ}$\yen$(~U5ýÚí~|Î??Ææ?Æ[]É
:u?Mo~ílË?þ$\div$$^1$qÖ$^1$ü¦?s0p?¯?$\times$_Ëåi¯
?ÃÉN¤¨$\yen$$\mathrm{\mu}$à.äñl+>.G?18?ÕØÓf«Ê[\?#¶Â¿ñSN{I&AÛÒ,½>ÌKþÿvãÁâí~?êâçðün< ↩
    ¾_=åÏ$^3$aÖ£C?ïè·ôuJxV¼?V\^0|?Ö OÀ?JÚÛnG?º½o?$\times$?w;ÄObñÝ$^2$i»èðäºøû?â0oÿ$ ↩
    \div$4õ¾ÿ$^2$ï¿yêaéa\ensuremath{\pm}?£Â???~¶¶g®xÜ®8ì\ensuremath{\pm}¤?ß¡eH|JÕ
 ?
‘UvS8Xnvåzq$^3$Øy<þ$\times$sÑqØ$^1$ìþÞ+®$\div$u?ML?½"?wJ?ÇÅ<ÈÊËæUWÅ\ensuremath{\pm ↩
    }àT,àpmè++øö$\mathrm{\mu}$BY  Å«¯äÛC0àÈ¾í¦Uï{?Éu$\div$ôn?^$\mathrm{\mu}$ÄÂÕÝw|? ↩
    ï¼Ã5¶Ã½????‘üø5ôu"?’?Â\ensuremath{\lnot}?:*i\textdegree{}?Þª»Ãöø?ju
UQ<)
ýx$^3$?ûÖðÞ]]\ý¿å$\times$fêï$\mathrm{\mu}$ÛbÂgZr??|ÔÁ
?á)¨Y6*«Zº??Óà£KkeVÚÅe]}$^2$¾-???*Ú??Ù?/#??{ÎÏówÖêõ.<=?zï½àÂËc¨å\ensuremath{\pm}: ↩
    ûâJRN*
J{?«Vq-$\yen$$^3$¤Qmf®dÌjªié*\ensuremath{\lnot},Õ?À\ensuremath{\lnot}«G4¡??6 ↩
    ÏÎBöedådhz¦?'ÃÙïø}>L?G?ûÓýû¡Äv??KN
p$\yen$\textdegree{}$\div$k¸&??md\textdegree{}J$\yen$©©VqjV À?PTt
$\times$Ü5\¯ É=$^2$&4?$\times$k§q$\times$$^1$êö|$\div$\ensuremath{\lnot}<^m|Ü=çQÛ3 ↩
    ,ôâ$\mathrm{\mu}$Sá+£&%xVâ?Eªj©lÃ\textdegree{}??p
º$^2$$^2$?ýDê-5ve\textdegree{}ÄÈÏaÜ$^1$óãý>/?åþ?¢ÿ#M¦~ã7Ò%Êr?ê??èÔ:Ù?<.?HVé?\ ↩
    ensuremath{\pm}?Âg$^3$BX$^2$¶nª?ñd©Ô,ö?ÁùÓ:$^1$Êº$\times$¸é]å\þw]NÎ?Á½_fäcÄÆÙ" ↩
    ýÔÕ
  ä~Ñ#AM^?[k:>?åÜ!(Cç%KYH§Ü´¡;Ukí?
¡û?wÂÞ§$^3$áóy{¸\ensuremath{\lnot}&&$\yen$õö
þ.¼]·???\ê~?@ââLÎ´K+?g]@?$^1$Kd\textdegree{}K4($\times$q?QT?zª¦¨)¨JËÞhùû?V?çs$^3 ↩
    $uâíÞõ»l;ý>Ó?xm?DpÌ?óçN$\yen$(}5&?$\div$T?gl$^2$Ýu7ö2$\mathrm{\mu}$ÁÔ?>?\PFN,* ↩
    N?3s7úÍ¶g'?ÍüeáúýLt®$^2$#<$\times$lù17ã<?6 U-ª¢[ð¨â  j??V$^2$
^?BjS$^1$(v??v4cDnË/
·¯?)?ü¯g¼ÃG$\times$æ½ô\ensuremath{\pm}uÑ«¨??OÆnÓÛE¿?+?v¸%«ì=ë?p}á,ç84éCA,Ò?ñ? ↩
    ÉÑFãØù¾v{\LF\ensuremath{\lnot}}lt1ñs=9y,rð$^2$ßiæ5m?y!i">9k?âJâ@Ö?þÏß@>}@aÑÔð?
ÄI&x¡?$\yen$?!??6?îC\Æ{\ensuremath{\lnot}0êz}=ÜíLLÝ{Üý<$^1$?Ïs$^3$?kyöÙæÀJ4?m#pÿ$ ↩
    ^2$?þ-W#2?Vt<_î?ÒÄ¤èéëÜÕ'¯?F^&v?Ý}\-< ?]-m?lõ$^3$u¢½õ$^2$"'6Æ?\ensuremath{\lnot ↩
    }F!\ensuremath{\pm}§áP?iU\ãê$\times$[Zsmh8\ZäÜ®G´V&ü??#?Iéw\textdegree{}åû? ↩
    ë¶a£$\div$G?/=»VlØééçåhmé@q#N>ÌfEÀz$^2$´i».0u_?&ÞÇû$\times$Ø.ÿ´¸$\mathrm{\mu}$
"D?èì'zãsucCÖÓÈÍÍÖg\ensuremath{\pm}¨ã3O
Ön¾Ö?Î?>î ?ÝÄ6m?$^1$Ç*à?+F%5%¸ð8*VY1?êÍ'Ý?1O'¼Z?YbºÒ~á4Â"1c~Ó5¦?6úw?@Ü5??sB¡>'\ ↩
    ensuremath{\pm}ÔCO$Õ?ÓîÙ$\times$|*ìþ1?-øgÓ£*R\ensuremath{\lnot}ò»$\times$#]î?Ïf ↩
    ?«OÖÃWÙ$\yen$ó9g¶Z$\mathrm{\mu}$$\times$?3¸â½?3Ó?Å?=IAÖ??jÔ6_?ÚºõvBW«$^3$$^2 ↩
    $¦2lrC&é¤Húìà´Èi£¢Ç1¦DËDæ4õãåegëåb$^2$ÇÓs\ensuremath{\lnot}$\div$r?ifÛ?F\ ↩
    ensuremath{\pm}ãÅVé??J ù=?vþ$\div$?5¿?Ê??
?¡æ'Ed~q»¢?5%®ÓQ·«'K?¯+?ÿ?/?Ç1
û>·?:\@?~WF*[PÓXWÕ-àVVZò?¡$\div$+B?~*?Ài?T&g(i<$^2$¢Ç?´l½$^1$>ËÎgF?^,?'¡ç?# ↩
    CØÓa¼HñÞ??2Á¤\textdegree{}£
LËáZòù\å«VÔ($\yen$®\textdegree{}
h?t'§Å??òÈ??Yðçðs¡BÇÅÄb¼?.??ï??Uï?ý?Í6?íì¿<,Sçi?RØÚYSñ~{k^o<?OCªJ¢8çàÚU<\ ↩
    ensuremath{\lnot}ãä  @~,øè ´tA®¦FÛ??7??<ßwÓw$\div$0¿»õlå1ÎÌzbQrIUÑ¤©$^3$^%\ ↩
    textdegree{}?$\yen$jÞ=oÁQÊ®Ztzu4Ç$¨ÚIGËtá'ò½u$^3$$^3$$^2$Û?òíÿÁqây{û$\ ↩
    div$åôz0fXÿ\kýãRÎ$\yen$¨\textdegree{}´©[ïrì¦GNÝJ~/À\textdegree{}ÀA¦ICJ@êÏÎÖ7'? ↩
    QÊIÊ43\?uö7©½ö?»øÜw?N·çÙë~Ý¿?#  fÊ!häÎ~lEaÇ$\div$yár,$^1$Ê]o;AÀå??sÕÂ?B¶ÆvzÔ? ↩
    ÓªTH>?hÒå´ØÓc{ä½»ïùã¯üy¼7>l,½F,ZÄ??r?¦Q<$^3$Ì´ u??*Ú
?!!¦?E&m:$\mathrm{\mu}$'§Ó@?º?pÐ$\times$+??cÇ?9Ö?Vç$^3$ÜéôzO$^3$£ü]\z<?$\div$¶?D:$ ↩
    \mathrm{\mu}$^?ÍÅ_P¶Ê&ñ%4åBO?x?DÑpÐñ\ensuremath{\lnot}(¦5%Tq?@ç6ÿÕyêºîÞ¤É?þ?/ ↩
    õáñÞ'k?Ú~
?UØV-å  Éáóz?RÐÑ??ÜäM-u??ü$\mathrm{\mu}$?$\div$?2@¾UööÜQ8VÏÌÒ´h.4ï®n>î¯ëöÌ$\ ↩
    div$úýû/7«6gc??íiÅ·ª$^3$\textdegree{}ä UÁ \ensuremath{\pm}"ëpÉs¤ï»Ú?§?= ↩
    Ë$äù½ÀÄçý3?ðñD´SSV.^¦v\ensuremath{\pm}qÑÿ]/?ë$\div$®Ç}Ì?/ÌQT%o+oh$\mathrm{\mu}$
ª?úøï};¾ÿw~wo?tñ?O?Öb´ýÞ?6ÎÙQá?qæ¼ëÞ8:\ensuremath{\pm}ÑÎg6V?ô3!\textdegree{}Ùa?º»W ↩
    ?ìï'ÞcA?{mI'§?Z'º©YðÃ<N\?úíuïpl;Ü|M¯??¶£HÎ
h
Ë?\textdegree{}@:nÌÑø?[Ý~?«ÃÙî?Ü ?ùÚjZ{?$\mathrm{\mu}$GÔ?<~ãæZyz?ÞL??2\textdegree ↩
    {}Ý3Ôeº'?'ª?"?P?¤?Eª,øZ@'^?îT$\mathrm{\mu}$¼nmo?>?? 9Ö.å1?L??î+]<.öNG??(Bh*U@? ↩
    ÑY?Ec$\times$}??.Oð¸Öþ¸ãA?»áK¸:?W?4fÃÍúä<w¡ÃGdó~£j{6?ìMFrÆ½{?$\times$víÂêe? ↩
    ¡O_ôq?(CQóòí?h]???7??Õë5¼-¶{>ÍÉb 3 $\yen$Þs\textdegree{}'?ØÅï Ú&ÀiÃ>bÇèçZHàÒ}6 ↩
    ÿÞËá£¢TÂ 4ØÔÈ¼ÑöoA\ensuremath{\pm}@M$^1$$\mathrm{\mu}$?yÛ¤D?$\times${?´CÀxwX] ↩
    cU¼Î?§§A$\div$ÚÐ&ÌðùCàºjÆúíÃÄ?Q\ensuremath{\pm}ÊÚÖa\textdegree{}6?.ø?fø?eØ$^2$8 ↩
    ?ñ? fH?%?á?
y)Ý?M!B©8YXû?$\div$ÔÆ#Dkü0lñbZZôè4C?Õb9Áü?C?ñ,xv?N?¦?Q@?\ensuremath{\pm}6$3"? ↩
    M¼Í8QUW?ÿ§IÄ?>9JÑï¨îB}änk%$^3$=gAÎ§£F$$^3$uE«Ozãb;m?CÓx-ÜÓ?<?ùfÌü6F?
```

```
}ñ¾ZF&àè?$\mathrm{\mu}$ªÚrÌ?©éâCHjPt?fÇÔvbd?:Á?c1g¸ß ?|+XÁ?&PËHÓ?Ã
?á)?»TîM6À\textdegree{}Ç$^3$Ö??Ó©¢?$^3$·?ô£Þ\ensuremath{\lnot}à??=^öû?©?Th?¸sú>?? ↩
    HJ ¦ÛÊ!/[+eé )???ÕÎ‘ýóg´?õ~4N?î(ç¸æ6ÀÙ% %J3$\mathrm{\mu}$¶¡oÃ?\ensuremath{\pm}? ↩
    $\yen$qY?¦pül1{?èÎBÇÌu¶?¾¾É)(Z"ùÌ
$^2$L%\?õvK$Ðn??QQMÐ
D‘RÏ»7$\mathrm{\mu}$£(Ñ?ÏÝû?äÓÓðô|$\times$ð~ý$\times$ö9§"ÐL"Å¼$^3$Ôfv¢??ïh?ï80ÅÚ ↩
    à8m.,?ê¨ë~  ?È·\textdegree{}¤~Ô«*???è?&Sôó~êÕ$\times$ðï.D?~?-ªÍÀ£$^2$åÂ?$\ ↩
    mathrm{\mu}$?Í
Ã ï?Ht:nQäD?9,är?Ü|û (9Îìs*so? näÕØ?·
Á?ë_?½é¡°$^1$.K?À>‘AÃBño?lcoÊõO
\ensuremath{\pm}??k D?û?\textdegree{}J?;¼?=v:$\div$$\times$QÀ??ú8?ÏAZ~!þñÈæÜä?ÀÂÀ{ ↩
    æÎU*A??öê\ensuremath{\lnot}??Ö2?§$\yen$æ¿ß$\yen$Úà?q$\mathrm{\mu}$Ø6"ÎJp¿´ÌzïÙ’ ↩
    ·\ensuremath{\lnot}?©?ì]Î©$\div$ã?5¿u$^3$àb^¶ÇÚPÛÙyë»ôMS(B?â?¡%KÌúé-5Þöï?FØ°? ↩
    àg??ÒOóÁ?ß[?ùÉÎÑ*?+ÞÆWSSÎ[‘?váx~%ÿ*á?ì\y®\ensuremath{\pm}XS?8D
ÖÒÕ?ò[[UÎ+S-ËPÚÔT$\yen$óeÝëz´§’¾HÎK~íCâ$?$\times$©ÔgÕÝ4\Ý9PbpYÑüsáM?Éë°ç.??Þ?{¾ac ↩
    %) ¨?ÅKgSÄ9å Å R))ÒQIO;?ÌÙ¨%Ä/½?ø©I\ýÛ^¶ìò\ú?½à  E9ã*iâ#\textdegree{}Ö$¸%6t?~J$ ↩
    ^2$l!8ÔÔ?pÀdçp¼ýÎ¦ÖAH¢YB?~/ç®êïÌ?<D?¶????t?GK?©m½ª?Y?h?>??~qçùJö+?)Þ·åøõ?õ?L$\ ↩
    mathrm{\mu}$¨)u$\mathrm{\mu}$&½$\div$br8íÍÙ3¿5¶?iåÿYm$\mathrm{\mu}$ÚO@FyÑv
/ßH$?\ensuremath{\lnot}ÿ?Æ?¨B¦YPXà^?J~?Àm?<\textdegree{}?ÚÚð  ?éÁ¨?ü.Ì~hæ=ÅØËnïV?? ↩
    E?äØX?åQ??B?Í>?H??($cgA?Apf:Ùéê*Dk_ýôúþ&0"¾c)ûð??K[0ï6@?,Ä^¤xùú?@?Óaàóå?¢U^Un]$ ↩
    \div$S$^3$ùýfÝ¸
ÿw\"ÆW¯ OÄúíí?GÄX\textdegree{}P?$\mathrm{\mu}$=OÇTæ?·¤h$$\div$òb{??.VçH}H?j««_Â_c7 ↩
    ?YO\ensuremath{\lnot}W^~yb~tßsiÑ@?XüSâuÔû|?új(?FÆE$\mathrm{\mu}$ª.OB{:\6«PMØðÛ¼ ↩
    <äk‘’R£c?l\ã ??Y>ÓñàÃ¡?û?~?Î@?Õ£¨=ÃÐëiF*$^3$?~F$\div$^TCÞ??~çvÑÐ1K?$\mathrm{\mu ↩
    }$Æt¾·G¿?\ensuremath{\pm}}"y?Eß+$\mathrm{\mu}$?N6ù:r?
Ü¡
?sò/???¤L??ç?ì|F½Ö\ensuremath{\pm}’úú??½»ÃG8áN5?¸BM‘?¦É0?E5,I{??.9?ÈdÁ©uyØíÁ~ ↩
    ï£Üðý·ã/¯åÁ£ Ê?ûPþ$\yen$å#??\textdegree{}aÀ?JËË¸~??É¼?Þ2û?îz_è?J$^3$íý½,®ü%´?è\ ↩
    textdegree{}|>?Õ£}Å ÒYÏáAxÐC´’PdÀI/½Ðg?SH?Útdö2;?6r9$^3$õQ?\ensuremath{\lnot}¤ ↩
    ,0d?À???Ú504ÃÈõ}Ú\ensuremath{\lnot}tÜØfì»¸ðG$\yen$Ù>?v  8}A8Q.E°ª$\mathrm{\mu} ↩
    $ËSÆ$^3$fÈ???è$\mathrm{\mu}$ 0??ÕòìæaioN
k¨Öç$\yen$´[z

Ileo?r5@?Yí?¤¤r]9Ò? %$^3$9æG§/~áa‘ã#Õ¼{6*\ÿÉ(lHñ?y¾(äù!(xõAØJ¨?Dh?¤?$B?$^2 ↩
    $4rÉ7dÐÃÝ$^1$PÝ??ûêRÎ J|ò¤??6 ????5?E&ü?åÉOÑNÎ2d®Ì?$\mathrm{\mu}$f]?ÑÅ£G?ñv ↩
    ~==4?HM%QMIX©?ÂÆG$?)\textdegree{}b8?‘|ÇPM?ÁÃ§¤¿çb?é?íkh5Þ?KW?¢Ì}ÑsdËFN??t’GLÔE ↩
    ?"øb?/Þì4Þ é¼?ñ7?íh¼#%\¸MáäŞÊIÛÏM?£?
ìhk?F E Í’TYhÛ{h¡G½´Ëe£øE í´Ù¾?8Éó$^2$Ð0?XG?*<QfxBeÈÐ?(????ásÓ‘M#DpÈ0~°è]èpâìâ·?( ↩
    ùoX¾(rcN:j)ô?¦?Â?¾?L¡ÒäM????[âCF8B#$^3$q‘ì$^1$xÑ}mÁGm»?Ôê9Ó?(
j4M"tá?ç¶
?©#E?p?8ØØ2#lêmGnòPÇÎãë½ß~’¼L¢@
$éâè?B~??eÅ? ???(?¡ï?LéI?È»"Ûc$3(Óo8»ä
¸Þ?¤@???Á£4*?rÀ?nD$Ç2\S‘?,~ ÔF  ?UùRe???îä¢?åÉ?,ûâ7eÆ$  ?
M??¡¡‘*+ÒM?Iù?é©rÇ?è??n??ÂPS¡Jpô?ÊA9?¼ÌÇ:Zh ?È?E
?Ù"½w?L
érÜ?ý~ác¾6Stáò
KÄ¸??jx¨?Ò´ÀÏÔñ$^3$æ%F4XdaÑ·?
P??©?ÇÏ’l
$^2$ÇÇ
ìq¤§?$\div$WYdL?+á½^Yi£
?Ê3Ó^ëJu???m)ù?g?
À*,Ï¼â)$^3$ï$\div$Ï?ë5Ö9#??lÖAQû{£jh$^2$=(?@¼ÞùÙTà?LIÑ?Í?ô?H8?hY??(Y8$^2$çÓF6®jIÙ[ ↩
    l¾?I?hQ?$^1$XÑäãG?´&¾q$^1$?Þc\ensuremath{\lnot}ÛáðóZÏÎt?îÁG  ØG?S?ììÑ ,?2nYe)P ↩
    ??RÅH?0z?Ða??ÿpËxcQ>?F~o>?ò/e7h8iÒå Á4¸?OO?Bf@ÁÅ%6??7òïDPy
<é ?¿öçë*B7?Aç?~ò?·1À<H\ensuremath{\pm}?-Ylé5
??ìmê$^1$z9‘
¶o
```

```
&ñdóHS??tOú;àSDHh?¤¢??sÆÏ?Ù!ÃÅo(\textdegree{}èÇ??<£èGH:1ï1Z?ÛÄ?^?Àù
c?2VpèÈ?Ý@&QC$É,ØWÐ#?:@è?_?hÊJ?Ô?ã À7æ2(??!DyHÆ7$ñ¤&k¼$^1$RûzÑ??B???u?ð$^3$@ÃÜ½~ ↩
    ú6ÖÀò[q&Êï$\yen$?Þ$8)Ò¼}? i$^1$?? gO?=.J?
¦???,?2E$è  Hx+Ç\textdegree{}
?\XÚÐ%??b&Ü=?=£ï??üÂ6OvQ1Ûf ;Ý?Z áI#?8tWOè??7íZl?QH,aÛÓd?~B?â3¤?ÁÔC$0'Çðä?# Wq5D( ↩
    áä·Ñ£?-$\mathrm{\mu}$?TvmkF;?X3QN%¾A?S¤£? DÑ?fJ»{3@0u %?¢;Q$ÜC/cE=?ÆÜ!Ç?~K&A#O ↩
    "^1#¨ÎË???Äil;,6ûNÀ
(Ñã?3\ensuremath{\pm}]þ|·Ór?íê??F??}??H??¢«\ensuremath{\lnot}?2Ò£:|)¸?G>ÓÔÜx) ↩
    Mæ7ìq¶6@óMmä¤o?'ÿ
*P\textdegree{}·?}tÑa?$^3$cJ?Úr¨ö*F8àïÅ_<Pdé'¶ó¯ÛHÛ½Ëõß1au©ÁîÿCÇ?!i*H&?ÚsJDÏõ$\ ↩
    mathrm{\mu}$¤?Ä'yÚ?* ?æ£t\ensuremath{\lnot}/aÅjøKæÇÈ
ó?À$\times$Ð\ensuremath{\pm}0æ>,RÀ=öG?¢Ät-¯&ó?è5?N?Ùy?Ìã/P?ûo ?Û©«s?Íxî5¢?ÑÇ?Ï ↩
    ?*???$^3$7SÍxî2ðý¸~ÂÌÐb~}7ÿt+éâ?aÃY?7dýBýý¿?Gd??ÁÛ#ùÉðòm?qÍ´
G5
Dýyj?ÓcÞt¦ÄÉéíÄ?ºÁ?\??e+??¦á?.?kþ·$\yen$wóqÓóíex´3Ô¿å©??ûEõ«ùõ?Y
?ðwQ»¾êj??ÉÚñ?N®æ,^kß$\div$\ensuremath{\lnot}Æ6)ý\ensuremath{\lnot},ùZ¿ØcÚÕèö.7 ↩
    LwòþÕ·??úåÛóSëþù©|Þ¼*ý¿?ÈÒÞ,LÆsë51ä®$\div$?w?(ªy
5^cóè¶k$\div$ü3TÁÐ&26|uÿ:c6EüÿxUHÅ$\mathrm{\mu}$îuÆä<®ØÏNd*ÑQ%Ì^?uÿ?:$\mathrm{\mu} ↩
    $ÖâclMÕ^ÜtxC2??ÖREöa?\½|$\div$ü?$^3$é_g?Vá?Öù}??(Rñ~7Z[ðé?'$\yen$§s6ß<SJÈ?itEú½ ↩
    ?õÊá)ë~?Õ?o@¶GEÌæÐ)b$\mathrm{\mu}$@?$\mathrm{\mu}$lÑ?iqømÂKwöô¤?*?5óàû?ç®\ ↩
    ensuremath{\lnot}?i@èú$\times$\ensuremath{\pm}}U¼?IØ;¦àûå
$\yen$â/]ñ¾?_çù?$\times$ìA9óoÝÞõ[syw@ìò[!?\ensuremath{\lnot}(?7ò<%?/?é$\mathrm{\mu ↩
    }$á~Ý?8y:~r?ÖFo@LÂõô"(*Þ?x«Û·ÌhÕ?F.?v?ò¶ï$^1$?$HhÚþ2ä$^3$x ?üÅü~ÙïIúôn?? ↩
    ®IuÁ_ùáømD©\textdegree{}?/Î"ôóàÂu¡f2?FZDBå?S®?ü?(Ô?ÔÛ?î?vüïÛÿ¿W?¿SÆÂ+Ï>~$\ ↩
    mathrm{\mu}$BûMìÓÑWÚ&}SÆ¨?ÊÿM$\yen$£L;»aÏïïö7å¨J?$nlæ
j½
:wÕKþ?uÜ$\div$ë¨_?C
>?'Á"ÇÀÕ$^1$æ/^?-ÎÀ¢Pïv\ensuremath{\pm}AàÒue$\mathrm{\mu}$e$^3$GS|?®ÞRf$^3$$^1$?f¾ ↩
    =O\ensuremath{\pm}lâ;ËÝ?{D?'îìÚÑý?éqÒ3~?Þø_jò?«ÑåN1Ú¿ÈVX)ûÖT4aÄòK:nûÀ?õî>ÿ$^2 ↩
    $yoÓÄô[t?{G4$\mathrm{\mu}$î|õAE:|<©=ËÓ?=ÌÇúSR?HñcùÂ#¨~\textdegree{}Õð=nwD$\ ↩
    times$Èi^s¡Óû?_+ðû?Z$å ?i}$\yen$À??>7¾Öaqx+ââÿÿþaÎ\textdegree{}$\div$ëvÑ?«?Sk: ↩
    ®È7ùº
¸&vÍõí?/¼éDÿ3p?+ùðÈ£m#Mrõ^p?|¼ã?Ác5aÐkÒ@ùÄ87_Qø¨sJ\textdegree{}å?.???\ensuremath{\ ↩
    lnot}¢¶æ¯«ÃÌÌäCìý¿ñXë®?s}cû^f9JýYõ¨ªÌ¸?$\mathrm{\mu}$êÿ~_¼h\textdegree{} ↩
    ÍÅJÃUkãsÎFØYû?ùäÄÜû?Ø?ðBaÒÀ,¿F?^yöí?ÓNm?UWé*?øÜ/{~?¸?Üü'»xVÆÏÇôW·\ensuremath{\ ↩
    pm}8?æù?/té?ý^?/5uE$\times$R-L?õCKÝ*^ø?E,?íFÔ?ÂóÚ?Ì½¨???fÛíÐ>?\ensuremath{\lnot ↩
    }âìYsËÛãÑmÏ6?h»º\ensuremath{\lnot}?©%z?ì?åc$^1$^Å? zÓ?Ü·?ìà?Æ?1ç(.«ü"tð`"/UH{?2 ↩
    ÿ??øË]$\times$oï?ÄwqÌº?óLpZBU /'4???$\div$©?&$\div$g3§\ensuremath{\pm}?§úD?õ ↩
    ??'?4m7~eïè¯´gªÝýî?1ÌØÒí9Isd$¾®Äd?"?ûÐ4r?%\ensuremath{\lnot}Zzw\textdegree{}É{ ↩
    XÌ?X©è%øm{?yÚx_~:?æ?·W5?f£®RÚ-vÛ4Àl,?\textdegree{}É $\times$Q§Ïñ_®Óñ?Îê29IH%ó? ↩
    zûZN?öæªrÝqö%Úî?$Ò]à£òèæa%!?<ký?Gb?ý£ÿú@?n?[0ZÀbª$\times$clÑ©A?^g¸?VícvÌ?zY?D~ ↩
    tQ}Ãn$^3$;?Ã?z:ñäNâ¿(bÓvÌú·¿W3sjË_îý?$}V?¿$\yen$z§B£?mWö¶a*E,Ö½V~)??"wlÍø\ ↩
    ensuremath{\pm}AA´ÛÂÉ¶@w,«oúA\ensuremath{\lnot}òw?Ru^ ËæWä?*_¦sÜ¶çÛXUR?iõ¯Ýáüº$ ↩
    ^1$ýC¸XnÑÝa?pÓ?õ?Ìüö'Mk§AÉ¶M\textdegree{}ÖçÐ?}¨6ËÓó¾_íÏ=8½\ensuremath{\lnot}#W¼ ↩
    ?Mm\ensuremath{\lnot}\ensuremath{\lnot}Î¡s¸{»$\times$\ensuremath{\lnot}:MHÈ?ÃK$ ↩
    ^3$ðú9t!kõ?ú?ßeÒ??M
ÿ~Båüg?W§?J:¦\ensuremath{\lnot}$^2$TT@??ÞC?ì?»y¾?¾È:'$\div$~Ó?'oÑ?Gë\§Å?zã?AY[Ï,T ↩
    ;?ÉWÖÐV+·)MÆWc6¯B"/Cûöè|YÀÊÔô?ñô½XÎ??æ£Éû?[ÛñÔýhæñÓÕò$^1$ü0&É]vfL~ã?ÿç8ÅKü?3A$\ ↩
    div$\ensuremath{\pm}sÉyz_ùÐ3ùã0õôÿ:T_£q?7ñü¢? P
?9|º~dÐ#??NU)#$^2$¡)?{$\div$eÃ??y?=_*;¸ó¸=l(û?L7»ÕK[¾^%?U|B~õï$\mathrm{\mu}$@5y? ↩
    ©ÏíçbnKî\ensuremath{\pm}¦$\times$~?ÿÚ> Âb?ü[$\mathrm{\mu}$êç.òãER
X4¤¶GfýÊïòA~ÕIÊu?{»?I?ë¦ºUû8ÙÑ¿~Z\þ?ìp2\ensuremath{\pm}Ü/?¯Þç?§cH~_øø¨©^[m?Ñðá?# ↩
    H¦ÄwÝd§ÏÒï¤Í¿?üÜmW£8á1<$^1$xiÃ$^3$ ½zòå%ü?+w]ÖÊdÀëÇ^¿J?Ûõb{?(ðý®í} ë }vêòW»»?¯?$ ↩
    \div$¿æ?*äÜ?ëÑìÃÐ ÑbBëí_æFo)$\div$òaO¡änîp=û$^2$¾^þÿ.¦????^¾Åò>?$\div$_ 1> ↩
    KbÌ¯Z¨$\mathrm{\mu}$ê9+5>\$^2$ÚÍ@ üÖú?ìù?iÔým¯»É??L¢Fë¡$^1$¿ó???£å\textdegree{} ↩
    òãÑy\ensuremath{\lnot}û¼)Ð¾ Ê
5ÊH$\div$(íE??©üCW? Æå
ÏGÍ)»åkï{#1çÄÛë\textdegree{}Î¢V?@¿ÐªN$
xZê$\div$$*QÛ}¨?\ensuremath{\lnot}ò$^2$øòÍl]BÚòn\ensuremath{\pm}ÙS$^2$Ûo?©}$\ ↩
    mathrm{\mu}$ßKº´¶?Fð+»Û  }¿??)$\mathrm{\mu}$b/&0eãù¿~o"ÄC}\??>
Æn¿?üP
»*ÛBl?ÏÃ6$\mathrm{\mu}$oö?
?¸Òoºa¯O?ËO?R,$UV?$\yen$h=OîPP$Å^ý?sðfÔ©4W
ä®^+¨kûô3òË'P$\mathrm{\mu}$CÆï8Ë4??G5á?Ëôål?ÕÈÎÄm$\div$zPv:"ïË¯;/??»þNMû~9Ë~?7ælÖ?
ªwf~uª@?Qs=»éÆl?EG{q¢ï$^3$xWW©ª?ÖG>vîÿ½ 2?¯ú0>á8í/ò½Ç&= Ö1?¯§ë¿è-3?~¤*?;iÊÈ/º?? ↩
    ÿnùÏíê@Á7äí?~$\mathrm{\mu}$?q¢ó=ÓXsq-J?Î?\ensuremath{\lnot}_óäb~·L¶k3xìç¸?ÕÔ0$ ↩
    ^3$î»y»_ã©\ensuremath{\lnot}$\times$$\mathrm{\mu}$\textdegree{}é
H%Ë;,¼?í???V2?F$^2$~FFqw!úÎ(vjÌ?õDZ?©i'=¸úÿï?õÿK?gþ!g¾QÁ£},?ýPv·$^2$3WkÁË_ÅÃÒ$^3 ↩
    $æÃ?JY£?Éñ?ÉEîÌxø-ÝÝj?Íêz¶  ã?$^3$s?;?@$\times$!y?ÿÛ\ensuremath{\pm}j¿9$\ ↩
    yen$ævlpãß=ù?Ái??ÀÖ'lNj¡BÞbÉ.õ:*¡wÈ]\§äË?ª$^1$ß$\yen$¡J:eî¿;¾/5Yã<®Z ?E?$\ ↩
    times$ªÕJK{Cà·|Sçà,Ë)úP?\ensuremath{\pm}Ø?©T0Ãø@}5\ensuremath{\lnot}$^2$?®Ñ*½?Å ↩
    '&å$\yen$ïia:_.pÅ§&ñäAo?Kka$^1$£êü¯ÎéýÚW6+ÑB~'Eöç?¯?ÎEºû3w?'V'$^3$$^1$!qT-N¤i; ↩
    Yü??Rok;ÅöÒ/Úê?5ùd®\Ç?Øó¸V  ???Ç\¶ïÆÜ,A}^?ÁãýÄ?ÌP(LóðÍ¯U?$\yen${?(ßÃÒÄÚÿà?l¾
ãgòa©wª+òøl?$Ãxj¢_zrx¾¯8+gºù¢/£ò??Ö´áÔu»?]? $ð=ð]jZ\úÔÔ<2»7Ê?âß?ýA;8|¯¡þ&ZÇ?Tñ¶U,æ ↩
    ?L?cÿ<9?Ð$\times$$Ã'áJ$\mathrm{\mu}$Ô/[??Æë$\ensuremath{\pm}Åð®ðbo¯Ä?kØ:$\ ↩
    mathrm{\mu}$X?ZMvéj¾BÌ?ÞbMöÅÄ<\ensuremath{\pm}~bnB>[C¼
F?þÆê?ëéÜ+$\mathrm{\mu}$AiÒ4$\mathrm{\mu}$wó¨~Ý?¯¼$\yen$õzõ U^És½Âüó»¡£$´Òh;Z?Ô.Ç\ ↩
    ÇËÙ$^1$?å?§ÒUÂ??+þ®X4Ü$\mathrm{\mu}$Ó¡¯=?lOÃì*óØx«?Âú?ÓïKð+$^1$Ò\ensuremath{\pm ↩
    }y=$\times$ùx¿C½ä?Ã|C¾¤Ð®\ensuremath{\lnot}kþ?~ÿÑÿ(_? $\div$ï8&>J} '$^1$?%3'$\ ↩
    times${»Ì$$^1$gT¼oëÑ¾\/ß¤?zà$\yen$w·ßºÚ¦}>P?Z?~<à@¾»¦ùb)B??$\yen$'»?´EÅ'?ú?bè' ↩
    m$^1$ãúý?ô??¯ãÜótB¿Çuxjå¡aoU/W?õfQç$\times$IùêÂÈÍ?|özÃ
_ìº6'®Ãùõz=¿ó*äø?«ë[7ùôa?D/?Â?ºõÀ?U¾ñ$\mathrm{\mu}$~¢ð\textdegree{}¾ÁÕ¡ø~-ÕWðÂ\ ↩
    ensuremath{\pm}Dá?Î«½gsí¸ç|Þ?¨W?þ¦?$\yen$$\div$ñìmb?ø$\div$¡O?Ë¿Ç®ooo?Å
:º?A?«d~©)5õÿá:G·F~}ä,nª?Ä?~-Áá=Ï$\mathrm{\mu}$JTÏ'$^2$?G?q\textdegree{}?GG@ÀÅiÌ?8 ↩
    ziFOÁ?Ë?î¸-]77?jnÍ7ÄÔ©?w65Mlòn?Öu?%æ.|?£naÊÉrv[¯l?ùX¸Ü?4/½<ZÜ2G\textdegree{}¶N© ↩
    ?á?è«h¦íþ5DHYa<éÏ<?5p?Ä?]¼0ÈnïO?3cdÐ$Æ??tÚù??äpä~º9éx'g}Õ¦^@åÓï5¤OZ?0)?æ?=6
?'S"?pÈÈ#L  eßKU??k?Ö{´ot½Qñ??4\ensuremath{\lnot}?ÍGÑÀ9?ìã´áTÄÜ¨å^Ø®wáRs7?l?Ù~;Ýmâ ↩
    -|ÿuãùè0~$FÜ7£j
2$\mathrm{\mu}$_c8$^3$\ensuremath{\lnot}Ê?BZDR??RÂ«8y,ý-i?0ñû ?"}W$\mathrm{\mu} ↩
    $ªÑÓaáC?}P«Æ jTÊ¤£\ensuremath{\lnot}?ÉDÁüÃW6ªÁX: ñìÀÅºÂÎhG|ÃÇÞóÞÀÇíÞ7ÒÅ????£A\F ↩
    ~é'8*Ó?ïF¾Ê*ÛOr©muñ4?ì@L??¾Mqø$^1$$\times$^\?õÛ?>d?ù¸^<ÛÔûèÈ?6?T6$\mathrm{\mu} ↩
    $K>?Hv$\yen$d?J¤?cã,BH©Õ  =ÇÍ¿ËÚM?ðjÙÎ?Åç\\Íí»¼3$\yen$¡ï?l?q??á*¤%?"ºps~0*hB?? ↩
     À?ÄA;¶Íå'ÍKï?¼Þg7M]z|Y9Ú?Æk?9¡9?Ï£¢S^ÅïÏ?ªvÝ:c#?6nxÜ¸¢¡??s:Ú'Ç~îÿmðê?ö? ↩
    Qæc¿5þÜ¨N5Éój^O????~Iõ??½6m|åE:i?g>Wäj¢\¢d??eÆ!)¼¸ä´î´\ensuremath{\pm}ßGêÜß¼? ↩
    õüxÅL?d$?óÒòhÛ?^õ-pâ%Ï$\times$?j
PUFBë74Îùð?¾Ï?ýôÜ] 'ò?G?Ï8q¾m??$\textdegree{} ???è$\mathrm{\mu}${ã#Ó?$^2$8pÅ$3?WN? ↩
    x?L$é$^3$öå?Ëé4=?á¦,¤?06ûùß?i$L2þ?N\textdegree{}«P?NpäÙ@UI?ZTèÄMÎï½öâ¶ª?4? ↩
    füËÜéyHê,Ø?£¦?$hGe8®?ú(Õ¤Ó?¿$?$\mathrm{\mu}$k;¸Bö(OÃ}dc??Pÿªaü£L¤)?O\ensuremath ↩
    {\lnot}æhZ?4ù?Lr5?Ýª! XL}i Ó2QztU?eê0?@3¿?ÓªY'Txg4?ØOXåÙ??Å$^1$^ªU¾%?aÝ?l?DaÝ$ ↩
    ^2$53?Z½?Ü3?4´:ÝQ$sR?é Zv¦//ýÕj[åAæ?!SzÏ'??'Zw?¡½èÇ$\yen$?31ÑÙëÏB$^1$h4ß%:?h) ↩
    ?>?$\times$+$^2$vH#òE?ïà?å ê5UãÔ¾?dnÿ?óÍÉYT?ËNÿÍ©$\mathrm{\mu}$?þeÓCÜjOÍUpSW´? ↩
    Ìü=F ¶¾-z}$\yen$Ïk?q¸Ü'ýù?)RÖ$^3$?´ÿa-À?ùoX^??âÙ_ÀÿLÚ;®_ûâ\ensuremath{\pm}N/7ô$ ↩
    \mathrm{\mu}$.  -d¡j\ensuremath{\lnot}=gMùò?õ}ÅOûú?"¸$\div$ÚZYÂ0t{|ü  ?#Ùíü-? ↩
    ZÄËõí?ìe=?Óúà?È??è%$^3$+Ú¯;Â¤ôSá-Â_smïï4]¿þ$\div$«\ensuremath{\pm}??$^3$$?àkZÀ, ↩
    åÉ?QpgØ?qbþXÕ;sÛÈ"&$\yen$ZÊçWdªä0¡?s¿LOü$^3$$\yen$Îþ·06\textdegree{}ÙâÅä?Ð? ↩
    ÐàÀX$^2$KbùÚ«lUn-Gô«2v6Xÿè´jö½Í?û
Ï%¾¾#oñyW$\times$úü?¼>$\div$þÑU?Ð1?´¦&?¢~?Üò4RK¦ô('#?sÖÐcY?}?bc·$\mathrm{\mu}$VÃo) ↩
    i?Zp¼?û¿??_itÚ$^2$?¡P½F·7½þ;\ßúQ7EÇ¿\textdegree{}î©ë?'?&??ÿ?p¸UiÀ?éËÅäa?{ì
Jý?véwÜ$8IXWÆw@ìrìa/!û8]ñ?o\?ãmFTM~Á#8OÝ+»?ü\Ø_??á  ¦?·Çwg¡êù}4jÞEÇÉÜù½K¼·%kê"»\ ↩
    ensuremath{\pm}ñ~ë»??åÇ+?Ó?ê^$\div$Ãã?ó$\div$jsø?Ç?VF?¢ÿ)ù?óù?x=åýÅ¿¿
ä?<AæÖHãæº·ñ0ØªM?N??è·
??ÿÜðïÂëpãKjËsú£!ñ:jW<|?¯$^2$»Ãõç:?9ÒîÂÝ?ø:Hý¾$\div$öë}?Í$^3$?
ïUÜðð?Ï1vÌØXÜoYìÖN%Å½ªË7ì?ÿ"Ù@!¸?=??ø?¯$\div$ËáÓjf¸?|nò,
¤$\div$?$\times$>ëÏ?¿?#H;ZûeÈQü  qÅ0Þ>?ãú?v¤'vIO@?1c½W[?
```

```
'?~?n?$E?U çA?\ensuremath{\lnot}ê:9?f$^3$ð?Ûp??ïõ?E'Cú$\mathrm{\mu}$)«Dm\ ↩
    textdegree{}À¿°\Ã?S
bü$\times$â °7?gÏ P=ó(?¡?Ô?ùO/
þ$^3$8mûé?<
?6þ¿dâ?~9«Ðd$\mathrm{\mu}$=$\yen$tØÝ\textdegree{}/~?|Eý~ëRÓ?\textdegree{}°JÍ)?$\ ↩
    div$màÚ
ÿ¢$\div$?èÅâõCýàÄjXn$^2$~?[Ów~Þ~4z|è&9êw
!sB?i
b%??âþ'?É
??ï9?û}£æw?æ£Ê?ÉØï?mé'lE¾þGpô ?ò§M{Ï¿Ã^oü?3RGª??$^3$F\textdegree{}??ÇÒfpP?ÆØ
ó5Óö,ïkQ?3mûB»[ü?»Gk Ã·?TAf+Tí
?öÇe«>R1OM0
b?E,\PÃÃ?u$^2$þ?iEï~Âtv?h®Ü!Þ???ÓM¸&Â¦'?ò'?A7Zi1$^2$æÁV?-?ØÐà?M(?¢ç?0?sì]p½?'8®Þs? ↩
    ÌÞ®°g??(y+Ô?ßod1):4)MæÒ?:J(dTî??u~!,??!JL«c?á?!=è?XXN\?Ðþ¸ ?x\Ý??Ñ??j«Ê?"Z§dr? ↩
    hðfi?X¾N???y-2ÌA+,íw?OÅ$\times$m®?Ð£ª:*còÌó?Sd¿ÍQYÄJ\ensuremath{\pm}U 'h§Í(-? ↩
    öû_6$
Ìß?Ï[mveû®½èýË$^3$??m¢Ø"A°vnvªwÚ?R$^1$-~Uõ¶bVÔr?IF?Ð?ì$^3$?"[ýìl¶yyWN$A?Yò\ ↩
    ensuremath{\lnot}5o©?Â8!>B¢*·[Äøy| Ú)î=*¶@?mA$^2$Èg\ensuremath{\pm}Ìò$\mathrm{\ ↩
    mu}$Ëõæx<$^3$?NÃ?Ò;Q
ËLZ?\ensuremath{\pm}?1<3~Ê?Ä??\r¼jªÄUÅä"VxE5<DBØ
¢á??Ó¦?cBËÇþ.®°þÈL_ïg$^2$Á7u¯ûÎ$^2$ÉÄv\ensuremath{\lnot}äõ-Mnfù\e´õ¶©URqý(?$\yen$, ↩
    r¢Ð$ðKDÖ???ûÅs«ÚìôúÝjÁs\ensuremath{\pm}ë{»?\q?è*·ö&?uÞI©Xs·fÁQmþ=äC¨Ñ?Æ®m·{ ↩
    ªâè¦Ãk_Fw8?q?<6lÕìþ0;Åq·òTð9'ù?ê°?]?·§%EË?æKh<?¢ú$\yen$ÇÐÓÏm)Çãñébk+gÖé¢ =Ò¤,? ↩
    $\div$ê~ônê*\ensuremath{\pm}¶$^2$Gc¨'Ö{d&£´ 958?K$^3$b$\mathrm{\mu}$qì[ ↩
    «·YIçÜt¿N¦¾ñ6Ì½.GRãé^øãlÇ?QÃ¯ªù§ê??$\times$5<dÎ7¦ðs$\yen$á?ÅàÎ8ç9É&Ã?´a$\ ↩
    div$Üyç¶ä$^2$ìr»Rò!pÀ©6õ????;????¶~""½$\div$ª«?sUY$Á£%É?Ym»[_[$\mathrm{\mu}$$\ ↩
    times$[ÞäÖÿ}vÿos¶Ûm¶ÏøÙ$$^2$H$\times$p?ª«øª"*???????Ó§Ov?'-$ÕÝ½?ìÔn?óo997]?ý ↩
    }~¯Oÿ??öÍ·??ËÚAåÜ#Ãé~CwÂÑým?Mu>9þRx¸ô:ÍÓ,Då2!#-??X}?>ßd2¼í?ññÙ2ÄýW$\div$C¤\ ↩
    ensuremath{\pm}u?®´~Î'z\9ý®?-ô$\times$Eö?¶STíØ\ensuremath{\pm}G¼ÁË6$\yen$"ãñð?' ↩
    Z\textdegree{}-§ä2fÅ©?i»+
$\times$Lc·Áz$\div$?ÿÁ?QFlâUJ
þ7.Âù?i¾?úó$\div$lXÞ¸ãõ£¶Ä$%?QÁ\textdegree{}_?'Ã?Âdgæ??!aÊ??ZX?  9o<é\textdegree ↩
    {}?qëg@¯kzý*e(\ensuremath{\pm}?ù???,B?Ñ?O?22ë2°ünÐ?$^1$ÿÉÅÎO4dTÁ?.4ËaAJ¢!]?=7 ↩
    vaï??C«?·ÞüÉÃ^Eý"'ÒV
*?î\ensuremath{\lnot}»?éüé<i?e?ær?í¦®9$\yen$?A¼ä\ensuremath{\pm}$\mathrm{\mu}$?Iæ  ↩
    ðqTÞZ?:>?@I$\yen$4t?QTX?6£®Ñö¼Þ*?Ñ|?(¼6uW?N??Û??[2þBßMbÊÊü¸W  ?¦»??Ñ?¨Â
9q?D?A7Ñ!Mö??ÆJ?aÎÜ }H?¸ÞB8ÒÓ?%+?i$^2$?{jÆØ6E?ÕM)E?*Icpzî $\yen$??:#ÔífNâvâkH1à7Îñ ↩
    >ÌÐß}óEïqY?È@Z$\mathrm{\mu}$!|Hf9I/c]?Â*H??VöÝ0ú½k$\yen$  íØ'?YEcÓ{0Äf&ÙÛ:¿$\ ↩
    times$½$\div$ø;$^2$?¯âkàu®ë¿?Êa@¤Ä7Ø,pÎ$^2$ÊêXzk\ensuremath{\pm}0ù_Ç$$\ ↩
    times$ça¯XÛÚw%¾0%?»+êÌùxßRï¡uÜÍÕüuõ|û9YÎÚ´Ñ$\times$Fd
9õãC|b¸þ[1õåââÙ1è?[{<$(q$^2$lRZ9Î{rìU?uN%?éA?ÝðÎÞØÕÓÍ$\div$ÿÏîîèwûÚÍ<ZîÞèjxkÓ,R´4 ↩
    }&AÚ\ensuremath{\pm}$^3$??ÎÉä£?é?Ñ©LU$^3$va¨·å©:,.°v$? 8?þ\^??ÓN$\times$O$^3 ↩
    $âðù?|?>
?
j·?j
?GA(*RIÂ¿
·èås½>w:ÿ$<?N
å$\div$?~¦Ðó??ÉÆ«Í\ensuremath{\lnot}¸,ëíd*/¿ä}$^3$¡òýz??½ôç|??~?ïáòjøæ4áÓ\ ↩
    textdegree{} %?¤
ÑJnjÜv]??l?xy,|Ðp?Ë?üßT6W<Ì?'l?ZZ?¦Uérø·>$\times$6ÕÓîì{½Ïñòý^îÿ¸-¶èl%-??,8???^? ↩
    çòØ$^3$éþ\ensuremath{\lnot}¯'¤c$>?r{YJFRpRtÙ?Jù?£$·_|»|9;ë\ensuremath{\pm}ï?y¯f ↩
    ?ÇÜø|?Z~B@m[Ë
¾¼zÖYAeÜ3Çöñ½¿3?ÆS,AÁ?¶òëzìV©R¦\¿S?õ,¼F½uàI%?ÖÙÕ?S?$\mathrm{\mu}$ñö?·Åõhý?Gõû{ ↩
    ÕÌ´Í¾3?íJ¡dT&?\ensuremath{\pm}/?¦?¨Ë?Ô?é\ensuremath{\lnot}u,L.?Uþ L?$$\yen$æ¡^ ↩
    Q\textdegree{}ù$\mathrm{\mu}$A\ensuremath{\lnot}nÏñ¡6?V»¿wçÍßãàÚmÙÌù>©p\$\ ↩
    mathrm{\mu}$ÚX$^1$FÝ¢zU£~$^3$4Ù]OEÈ.ð?Ùâ^d\ensuremath{\pm}Äé\ensuremath{\pm} ↩
    ôoQsíÕ=?©lÝÙQ>ÿ»îì´ú~¯$\mathrm{\mu}$Øøstö;Þ?»¿§óÅ$\yen$àÒÖU?ð"Ë?ú\ensuremath{\ ↩
    pm}:)\ensuremath{\lnot}Ð¢bÇ?+‘nòñº\ensuremath{\lnot}:8?kuò¼V?ð??GÇ?«§N¸4? ↩
    êóèãþýÿ·Øû;Í?æ|Ü;Eä??ÌÝ\ensuremath{\lnot}
%¢p®\textdegree{}p$\times$ó?ß]rúÁôØ$^2$eyüú(p?d-ÈÆFnq[&??ÚÌ?$\mathrm{\mu}$t?Oòc_? ↩
    y¿üýß’iû{n»»ïßjÈ%$^2$uÊ??æ]+\{ ¦¶D9Þb~¿Sþ0}?¦A0Ç¸aq|;?M9h~nê6!ì=?Þ- ↩
    Æú½¦ÎßÏ¢ßuî£º@V$^3$?kÀ?TDK?Ç-<PÍ@?Abô$^2$:¸<ÖLð1ñnÔ??ø?<rBUSHÛN<?hí¾MzöZç}?¿[ ↩
    ~ºmºHäÙHÔ?$¢ªÊm?P?7ëò9\ensuremath{\lnot}$^2$ºÞE¡Òº^\textdegree{}´øÐ~ðÐù\ ↩
    ensuremath{\lnot}ªc?(ÕI?T|xº\ensuremath{\pm}aÀñ4ú´¿·Çã«%Ø?$???v$\times$Õ? ↩
    ÇBzÄñÒÖ  Ë/©uÔÊô¿?áÌcwQ??#Å·Á»6}?BK|,cÇ\tÓIó§ptü-{}É~öK·$^2$?ÞâIºuñ? 6 ?ÜÀò? ↩
    ÁcË£Ôõº YLúìüôéXL)^ç??§ÖÁÆ^?HUa?ì¨3?ÍÓ=¿??ÆÃH?J
?Y+º??\ensuremath{\lnot}x&çS É{Mÿ¯]/,6X·uÆr%*çw´ }½$\div$î~W%GÞ?:Úu5X1?¢&Ô?-Î$\ ↩
    times$$\times$ÑÜÃYñIÂ$\times$Ñ5w7o?Q7FÄìhð4?/?ËÖý<Ï7¤~0<-3ó=ÿ?7e\ensuremath{\ ↩
    lnot}bTÐ[ ?#Åoo
\ensuremath{\lnot}UNúÀ?kUËú§ÉQ¡?j"êj?mC~?ñçV?Å¶½??næ?íÉbÇ!kûA¡BjïmÖç?Çq?¾B)3§MX§?\ ↩
    textdegree{}º?L?Ç?ßÓ»?_L<?EÖÖj¨!#:  ·?¨Úàõ«È?cÊéå_u$^1$ËíV?"(Û??~,m$\mathrm{\mu ↩
    }$)UV¤ÆÂT¤Ë$^1$5w%$^1$ÞÏíx´?ù(G¼úÒXCysèäá¢Pz*#cì$^2$Ëõú?\ensuremath{\pm}îö\ ↩
    ensuremath{\pm}6u?Y|Z9/Ãý+àÍògUHª2aT\ensuremath{\pm}%BfÆf½ö6ÚwãV"+ÇÔ?:?"~É$‘Àw ↩
    ?  Ò\& ©Óæ‘*cèùØ$\div$VÛ{Âæi
<Þ$\div$«ÀKRÜ?è©~?s¨4À’Àß?ÂÑ»Õ\<>¦j3Å$\times$-ÁóCUkph?$%!
$^1$GîoS\z|æ9]? ô-Ã´)?eGuÅPb£ß?)"?À[Èê+?þ?ó?Mçô&¡ª?Uå*L4JO?¨|Iêº?ÐÆ?Ö?[l\ ↩
    ensuremath{\lnot}\O?-H¡1ÞOÁ«}¦Ö?&ï&bÍ(GC·6?QÝñNÚûZø3~¯?«
]À!F:?Á?5iÅ\textdegree{}¦hòÐ?]Ùßþ¶wÝ<?IC*$^1$|ßM?CUýhÕ¢¶ómÌ¶\ensuremath{\pm}*: ↩
    BDgÙúzùÍÛq´oÝÖ?cçtêEUgjq5ÒpÃÆ0½
‘ä~$^2$?ö¿Xâ§äÙóñã-?ÅÂuW?‘Å.!\textdegree{}ìd?¿(\ensuremath{\lnot}í-¦ÙÙ®voãýþý?æ ↩
    ?~???u?Ü?8WLní?_?h?½Ö?EÓ?.?\ensuremath{\pm}çà‘\textdegree{}?ûËqÃÀSH&M?ì?Ûi¾Ö$\ ↩
    times$$\times$í5Ø$\times$ÕÙÛÛíhi¾{Äún?(ë?¶DH<Exß¯¦à4HV$\yen$?
z?Á¯\>ð0-ùøjñ\ensuremath{\lnot}*ªlÊ\textdegree{}cj$\mathrm{\mu}$ÏÝñx?êø¶77§ééñD~( ↩
    Ð¢Láü  é8?ÑÐ¶::QÖ~ÊÑ%øAòÙYÜ$\times$%K?UöÙLWad‘©$^2$?s
J?«öÚ9Í<Y¾6$^1$Ú?ÍtÚmèi5Ôð<?»ä4?,¢Î¸¾?lamÅÞ
¦ KËÚn‘dÙÀ¶?Úÿ®??1~o)?a?^?Õ?£ÚiàuÞÍðýVsM®7?Ðí$^3$Å"?É?£?k;+üãAx
å ß :þã§yÒå3óÿ^?V8l$^2$9¶Ò‘Áï8ÜÑú$^3$=ÿçþ¾ß$^3$$^3$ïÿ$\div$3ñïjéññmíN#X«áN¤5\ ↩
    ensuremath{\lnot}?¡Mg]©\£èlíì$^2$?¿JâÓ/?<L,[ªÆÔ6A?ÆÜJ?mMÅ®Û;5¦¯»xó>?¦fòÔlæpòI?‘ ↩
    ¤Õº~ã§=dÇcogÎ[u~È?{îk?ØAó???{v6Ø Lòè?vF$\div$?íþÞLîÏy¦$\times$øjù7a¼u9ËTRl~þ
¼ð@&ZéÏ7#ù{]?ÅvõB,Åu?]kü¦XLaa*^rd?îïÏójgø\ëCÅ?Í9?z?Ý$\times$».$\yen$?A
(
böb©?}§_©Óçz?ÏãxÕ½$)nQÔþ>¯[«Ì
7¤~y5¿
$^2$+ïV?Ì(¤ìhçh¶oHÁ AcÓ9!bÜÏ{üVØd+ÃGú?qG??9»?ºL$^3$gÿ?ÙûlïdÆÖÌãV?pðQ½Ó5¦ogð{\ ↩
    ensuremath{\lnot}ØPÎf¾ãF:5ÿçÝð?9¿§é<\ensuremath{\lnot}¢ÝÿýMþ6$^3$?Ôôñ:\Ó?_gðù?´ ↩
    ^¾ñ*T?jÑàú?$\times$ABÙb#¨?ã?uF/ÃîÐ~©ÂèdQ.?$^3$o;üfx}ýëÏßë~Æ\öA?Wv?w[R?®k,,?¤?%] ↩
    ñðl4ì¼?cd?\textdegree{}$\yen$?B{|8O´$\div$½âVG$^1$ìm î)ªïsºÒ~{LV~Æ?[¡Ìº?àÒ?.!* ↩
    ZÝª??$\times$»½?l I¦R¡ÓãÅ¿èY$½$\div$?ºå\textdegree{}K’7òâ?Þ|Ä «éóý~ÒSÅ§GR?F‘/ ↩
    ¸º\ub?0«klÎJ‘ÀGÅBmùNõm  1KñüÅÕ¤[iS$^1$Ø?ÿ7õ?!1YV]lk:ín:¶)ËL£<?\ensuremath{\lnot ↩
    }?ý3ï7M¤?a> ‘
\textdegree{}¨é?¶ö;ýÅÛöÄp<Í?$\div$»§käçÀÔ@G?â[Õt?NÐÝ?;Æ?Ú¦T$\times$oXa£-$\ ↩
    ensuremath{\lnot}ñÕ?=:$\times$ÌÐ$^1$¯s§$^1$ö0aü¼ÖwÑ©åùðó8Úø2®Mð¡/? m?)ëD?V$\ ↩
    times$Üÿ]+O0<UÀgÇ\ensuremath{\lnot}§¢?Ôlk?T¿ßý?$\mathrm{\mu}$Êöq2ºa,$^2$,? ↩
    ·Acíåá*??öÀÐ?Èi?q?TÏÜq©ÄDÂ?Â$^1$
Ø?§ :·ïf$\yen$f?L/¯úF$\times$FÏ§zóf\textdegree{}Ù$^3$å?$\times$Ú0
¾C?0$\mathrm{\mu}$$\yen$,ÑHR·´%HÞâ?U3gkÎ7mÎ/áÑyý½Ý[$\mathrm{\mu}$Hã$^3$6;¿íî>Óéz, ↩
    òp=\ensuremath{\lnot}¢iæþ?]À¶d$^1$´%cRÕñj6Ñn1,?P¶9[nK‘f_ÁjÚö¾t¤<íd?ÿ~$\div$$^1 ↩
    $ñ%½¯F»}è?l?@4é½Kéz¿$^1$«o¯B£$?$\yen$ê?$^2$t?!
*åYÒ¨Lÿ0ú2®®1CÂ¿»5[}ÿT"Ù¾$\div$ÒÁ½¶_9d
ä«*14PvÜ\®O)??yßØc$\mathrm{\mu}$?8?Ì#A¡S-ûÙ¿ýø5/?\ensuremath{\pm}cràä~c?ºO?9úÙú??? ↩
    »Á=íþ?<;À?FÎ§ÍóÃL?)I\ensuremath{\pm}öà)Y?ª?c$\times$Xlæw½üôn¦ãõ¡=4tÃ¾vb\T!ûLùÜ? ↩
    wÜÙ?%^AjpDK7$^1$$\div$ýÕ2Ybtn?]û£V\ensuremath{\pm}ód\*âCö;otË.Û(?"Ok$^3$ÄÐ6^Ë?© ↩
    =k?¾Ç$^1$üÅÉ¯Áã¡/ò®þ·?$^2$ÿ_$\mathrm{\mu}$çª$\yen$£òB*?&}?HôÇ¿®wl;Ëì~ÕØÉÔú?¨ ↩
    oEózÒe|{ÿÞïót[ÐgüÍïêíÉ¿B ?uXà®$\mathrm{\mu}$?C$^2$ÜÖ?ñõ0.0$^3$>=¼½,?<îÝ?n?K½Àî ↩
    {/Ú?øxaÓCÏ(zýM?5Ó0çj}Í’D?$\mathrm{\mu}$|?)bZz?Åë1AB«OÚ{Ä>Çéê_ãá:dtö5A½À][ ↩
    ÇªÙ©5H3ôæOsCtÞ’7?¼*Ð?F$\mathrm{\mu}$ÙxËË?$\yen$Ô?{xùø;xå~Ü\textdegree{}
qÎÎÍþôùJmú$\yen$häíjK?$ª,®}?Zü???lÙú?§£-?À=®úúøièùm«ø·æçÿÿøÉùÐ?èfåu?$^3$a·ðûq?~ ↩
    RjØÞ^ùÊ%¯wùÔQÌgÏ~Ó_mÇn?ü¶lüÐ!|~ïÀÉ¦Ït\textdegree{}>?$\div$?QÏE£çû½ïçÿÍÒ@F~"?J! ↩
    ÐÅÔÿ^ï‘?ùÌÙ\ensuremath{\pm}íýÏwoØolvï?$\div$ýËI?^Ã<®V1Ðb»òAV%X?á{¿òü»ôPÙ?#ü| ↩
    pÝE¸N7Þû??Åÿ;ïË?MÇ$^3$y?¨8m&åCWuíß1??ËÀØ_v®Ç«¨ë®1Å?óÏ$õ§#Iqøï6z-ï^ðãÇ$^1$ÆÓ$\ ↩
    yen$Õt?K\ensuremath{\lnot}Êå¯È$\times$üI_\ensuremath{\pm}|ë+jÀl?©õS{´¿òÎzZñ}7?% ↩
    U0lm-/FYske3cû{?Èý^«<»?4\zN>?s4*þ?gÐ?¿ö»ª#®0ù$\times$þ¯$\yen$}$^3$ööiÆ] ↩
    ÐdncÍÚdÃñ|»rªX§´?õÖëm_ëîL[\Ù?Ç?cìÍ:åTÎÕãOëüaWûÐéôñMÐâhÓ¿ª?$\yen$ÅêëèzM¤èlp
\ensuremath{\lnot}?a^$\yen$¨ïÚ.Ó?¼7[ø$^3$ætØw¸v?~w1??j]ßùû¤?SÏ???ØñÎ$\yen$?é5$\ ↩
    div$xiÜgx´6ä#!MQ!i£$\times$g7??öÇ?ÃôI
å?$\div$$^2$ÃèuÙà’):z$\mathrm{\mu}$?¤’@êu?ßÌØÕm?Qâ?ÔEÙ|½
&V*f$^3$S¦Jn$\div$?è?ô½ezÓþ_éî?_??sÃ ¦¦¼?ñYã$?æË]Éb¡)mn©?zÜ|p~Âåñuª¾?q«ceK?íé ôý$\ ↩
    mathrm{\mu}$|Õ\Ä?» ¨Èëe\ensuremath{\pm}%5=?XøVØ
L§‘I?ÚÀ
J)çgÅ¡XaÆ3~gõîJªÓ
& hºüj?ÓgûÿS+hbÀø|%?õ}|¼¾r9Îg7~úDd·,$^2$-|Â$^2$?ímÕ4X?¡ñ$\mathrm{\mu}$Õ?O½,?VK¸iÁ? ↩
    C½ØÝ???àÏÚ¶5íR5\ensuremath{\lnot}Ø¼éoýö4??sÓå’Á ñ[¾G®?
ÿ?ð?PÜçn9"h?’I¿{%Þ«Á;ÝÉÜO¡el\textdegree{}¡Ô¤¦?OFE@M)¼Â$^2$R?ÏÜÊ?¾ðwþðb?¼txv§ï?W$\ ↩
    times$ËèÝ£}?NÐ?Áy$\div$å?OvOVÚtÁ¤·2NH¨?dÛ«>$ÝÙ5Ü~8«,LÀì)ò¼?-+’Áeê$^2$·\ ↩
    ensuremath{\pm}?Ê/uyû¼ã
uü:Ñ*Yù#?-[qøõ#?X £$^2$5_½£$^1$,qb?¨4ÇeSm»§?Ò?¨yêQý#?ã£ñ¿£æã$^3$Ë½^qô%bÒ<®óÏ¸ió7% ↩
    q½cÁ~í®~¤ó $\times$%&ögksÇ(z¢Ä¡\ÂçI¨VJÒeG2ð7½?Þc¾?Ü)}\l\textdegree{}ØW¿øº?t’Äçç ↩
    ,Y7c9qCB?ÕÌJ{ÉÔâx¦É"¢Ð<?gc>ËªÊ?\textdegree{}¿@$G?’2S‘vØâ‘Jx;QCÂËÿlD\ensuremath ↩
    {\lnot}:£ì&^½pï$^3$?þ|(áÙÜÌX?§K?vÛaÜxÛï:ÿnÛKw%ëËíÍÝÕhùÛæ|UJ/?_FaEQ»?½?âÏ$\ ↩
    div$$^3$é1¾d?eÔè?§ÐgûºÀhÊV}‘? É½ââßqAýIºßSLß¯òiñ}?ß?üwÙù$^1$Ü??N&$^3$Oö? ↩
    ©ÑÆ´eëywë|?ª1¯¼$^1$}N?_:¼@$\yen$$\yen$+ýÝ&Y?ÝÂ?\{ËÐfÖ.??¾N7ú?øü{Y>åöý]¿$^3$ÃÀ($ ↩
    ^2$¦½i<A¡p9Ý?pLroÙÞ?p>kÐYt+ñ?f$\div$?ú¡®;{ìåtg¸ðïE£cE?aé)áÐÓù36Ê(ID?[{í)ê~??O¾¯ ↩
    =(~nîä?E\ensuremath{\lnot}Â$^1$¼[OäúJ?õeûXln/1ØÍ6e?ÙLý¿¢¦P$^1$ß&~9?Ù>®IS§ÂÖû| ↩
    M43%?À?Ð¿.Ç$^1$ÛÎ?;^(?A/1¾sLí1ÃõÃ’KúæRÃý~ÏD5Ö+pØ"r_A[?Híã·*Ü?Æ2¿oùdó©ã·’ ↩
    gwsðí¶jýlm$^1$£Fô¢?åþüE-?]ª?·NAì|ÅÂÉõqì%+u$\times$Nt$#?û:09W?m?M&øJ?á.»7·âiùÓ? ↩
    Íù¨¯íQñç}?#G?^I~???Ïiø}Á$^2$2q~5?Wi?^?-Àæ$^3$Ë¾·¿??$^2$ÇkË?{4Íó=ï¼ä·¼þxt? ↩
    ·ÖÕÍlYÙ$\yen$x?YA?}õçÄIXlæèÔñ{ÿ
0Jô½F!_2P$^3$ó|mDdÏÛô0?$^1$hÓ»@ñì¼^ÇY^??¸Ýù>çâ+ÏÂêk"$\div$Kqq?«xéüñ?Ãxs\textdegree ↩
    {}??$^1$ZhtÅË?KiôM:ïÞæàÙE?¨Bñoï?q[?ïOâo?$^3$\Î5ë$^2$yLÙÜÔÂþ?$\times$‘8û+/ºá¯Ïòý ↩
    ?Óé?¤»ÏÇ?n8§1´?\ú??k?$^3$¢½?È$\div$
OÕ[ ^$¡Ã)?*^?Í·ROGÎËÆóí?Þíë¸f’ëË½ÖÚø½ßyê\textdegree{}$\times$V+uuqÉÜûá}¿r^¶^?ÿ%?? ↩
    dà¿ü>?Ù¿¦O_MÌ)???ÇU7*ò;^Þnu?VW¼ÔÅP|èCGoÊhã\textdegree{}àå$\times$$\mathrm{\mu} ↩
    $Þ]Dd|Þ\ensuremath{\pm}? Ç;??$\mathrm{\mu}$è)I?$\yen$èQiëücØþ(?Üy8\ensuremath{\ ↩
    lnot}HUuwÛ9_ÍHës<?yW¡üÃÂô§Unè~+$\mathrm{\mu}$?ÿ¯tÈwõÇÕYòfq\ensuremath{\pm}Iø¸w
ó=Kì\ ®Éx$\div$q¤?¡^á«©®ä~?Æ6?Âs{ù½Qsý/ßê¯?%%Á?!û,lúök¿ß¿ó??ö>»$Vc?ÿ7<’m o.ÀØuI¯$\ ↩
    div$i¯¨5$\mathrm{\mu}$$^2$oõxw8a??R\«®;¿?ÇQGÉ¿^L?\ensuremath{\pm}Z5pqwFeVyñ{ ↩
    ¿XéCóYZ\ensuremath{\lnot}½?¸ú$^1$?¢Åçõ$^2$?:ôw76§?
PÕXU$^3$$\yen$?y¼ ÓØs??\@Øs#?$^1$?íyí%î?¾èc?ê»ì½$\times$þ¿Îm¡¦]l®pÕoØÌÞoÁidÜ?ÆË?? ↩
    c¶Ä ?@ð’»uV’$\times$öÿãàórºaÖyÄH!8uF6WI2bÅ£Ð?¾¦§cí¢ÊÙ~OM?óU8_níÑºAÎçß?§AÎv§5$ß? ↩
    Q*.* ÝÍì8§kÇqÙ: $\mathrm{\mu}$?;\ensuremath{\lnot};ï;?9d´7ÞQ(??K%}#_½¼?LÙ?Ù$\ ↩
    div$<QE?.råæ|?ÝèoRÎ*f)$*»?ãÝòq??S>?ÕEÕÁ»?Å??[ÏÄ¿Pïÿëþ?\ mæs?ÜÚÉÙkñ~äûÖ^·P4PtÊHá ↩
    ?\ensuremath{\pm}îz\ensuremath{\lnot}W?¯  ?$ô¦ÈÍÿ>*?i~IÌán$\div$lÀ,q+öÕÂ¨p? ↩
    íö·3quj?m]øÄVÜáÔÿ¿¦
àÙt,ÂrÓéBm8[ÆW7öW¸£?3ÓÂ??NW;\ensuremath{\lnot}?õe=$\times$=Æäå?¯X"?wga«Y\RM?\ ↩
    textdegree{}VxÔî?ZoD&
}??çcBQuY9?kK?srN)aà)ÈóòÙU6ÑÝÙ/MWùtnU?t$\times$Å´ ´Â\¦Å?öÿ_ûï´ÙÕñ?Eû?ò3~?ã_!$\ ↩
    mathrm{\mu}$Þpk?Ë>NF¯õùõC¿®>+À8¤ EA?_§ìu-öó¿Çú$\times$¢¨<\ensuremath{\lnot}<? ↩
      c$\times$Q»h;Æm¯\ensuremath{\lnot} \textdegree{}¯ÓæÎmTº»5&=ÙñÝvm§/ÁEÉ©åö~ ↩
    §ìn½~ÕåÍi??{<ë?_·|}/ëóïÄ[æ$\times$¶ú&di$\div$Ú·jz»_·ü/?\ensuremath{\lnot}zÓ?@|2 ↩
    òöÚiëéJK}f$\mathrm{\mu}$?F?§?roÞkoèuÙäòcmêÊª½ß.ãÁÈ$\yen$^?_$\div$]?X|?¤u$\ ↩
    mathrm{\mu}$4|ZCXn\í,ý
```

```
7;Ü-ÏS?ùÊÔø·?k¼L©?6??‘õåÎWëçS¢âç?å)ÊGë$\div$c
Mr‘¦æyìPÈ0®Õp?:k"©ÿ,´ù¾M}Ý\ï?v~Ä£IïMw?k[X"È?]h?ÒK?Z5§étò½¿aZ{pî9HË+ª?.}Ø\y\ ↩
    ensuremath{\pm}N?¯?5?EÍ^YÛ:??  4a?Ákä®J\textdegree{}£F$^1$4??öNª S,?ÌV?’8úÿ?Üì¾ ↩
    ^U¸H?Ý\ensuremath{\lnot}\èK®=®?m¯©Û\ensuremath{\pm}]QÜôKjö~~ü$\yen$)¾¦\ ↩
    ensuremath{\lnot}7Î¿¦?Û?d(Íw«ÃÄ?rà\ensuremath{\pm}À"MS?4ó;óïò?wS?Ì?eZLú?k$^1 ↩
    $Þ¯êYÈiÝá\ensuremath{\lnot}?~ÑV:?þ¼ï?Õ?R?IÊ£?ô’~??&Ù%a?$^2$?Ë~?4E¤ø?Â¸7WêÿS/ ↩
    lýÝ$\times$¼(9ð.p?TBý½n<=|îÆ?Ém$^1$·wë?[?¿Ùã>\textdegree{}?u£«$^3$½ñ~"?ôå[lÆ~ ↩
    ÕÊ$e«$\mathrm{\mu}$ÜÊ?ÿþk£@«+Úgíâ[1$\mathrm{\mu}$¢ø\ensuremath{\pm}ä!äóý?!àÌü¶æ ↩
    +æt+ÄÌ,\ensuremath{\pm}~ëhvûPÚâÞ?Ý$\times$iãÏs?ìuv*Â\ensuremath{\lnot}â??(R+6? ↩
    C¿g~CÛåóXØß$^2$?ÀÜÝzÙlïÎOaLÃ½$\times$}$^1$ÞWH$\times$?C;]þëY?OJ$^1$IØ[º¿?zgÑ?P ↩
    \ensuremath{\lnot}Øô??(ôÒ?~p9Ú?¢*?$^3$?k?äólB  ?*Ø2¯3]*¦+ºíîvé5öþk?Ð\ë$\mathrm ↩
    {\mu}$ jô^\ensuremath{\pm}¨$^2$øó\<?zP,\ensuremath{\lnot}?bK2eÇâÐú,nRËÎ6KÊ
N?$^2$ô¡¯¾VdÂ:òæL?3é?GWp©eÉd$\yen$#0ÐóKÜ?ý´?$\mathrm{\mu}$?r!’?PQV0nÿúü&? ↩
    såÌcMê¯CÙÊö¿Jó?Á(JtK$\times$ áHÓÕsPbeçW»?©0PFâ{$^3$¿0¡8?§=w,bIV¾;3u^R»Ü;ÄGð‘ ↩
    W2êû}No$\yen$bÎ-ÀL?%k¡.
c½VÏ«ÒÔ12p*vÙÿ?´iÄK¿Ù(©´Çå:[ÿÝÞ+9Øö:jBÁÂ£=  j$\times$7aÛ]‘\ensuremath{\pm}®ú»¨Fy9 ↩
    ?îo?ÓÅ¢?’ÕN?>?gb?ÿ]ô¨?½9-j?$\times$·ïf<º?ðÖá¾
kóºåàu$^1$)q<û~?ñ9/$^3$¶Õ)Sfí\ensuremath{\lnot}»$\yen$\ensuremath{\pm}¦ýÜ?:?u?}"dA ↩
    "?$\yen$*p^]ýiðÆð{ÿ(A?ZvÊþÝwxRyòF$\times$Y-´Ææs
DÛÓ$^3$Øj}3?ÓzuØó|?w?É©ðyaN$\yen$?ëÙ??£J¤?R«¨?cJøþÝ»+í®0*Î‘§?"F Ì\_3?8Èç}ÛU¢? ↩
    WÏZÀñ9o5øeE·?5V6?bîç5!üjÄÓÝ/Mr.Ç?ß$\times$ÙÔBjÍhw¸ÉN#?<j$\mathrm{\mu}$Hqûr/®C? ↩
    þ¼  O?¢b?’?ù-?q&où3å 7DÝN2Å!q½hðÂA+RÅT©ÃÛúô7Á\ensuremath{\lnot}\pNÚ?Î?&PIÂ<Z> ↩
    ZS$^1$£+??ÐÜPP\textdegree{}?SORn8ó(L?UGñ+ÀÃ-ÎËª?¸âÊ¨Ny?tûgA=nÁ?¶?]këJ?JÑç@L·+ ↩
    ÍÆd¦\ensuremath{\lnot}*
çP1X?VV"í¿å®ûh R#Í}É4cê&je")@*?|?#,rrú+!^ûBauäõ0
?-Æô´8Áa~?2Q·$\mathrm{\mu}$ûÿß¡à  !a??Mí§Û-,:À?Za~??¿9quêí/Vêè~¶å$^2$äÝ??©’zZ©
Æý*å?6ü{ùút·$\div$Þ&D·.â6rÛ}ø?&,P¢C?  Ø~½å*?4 ?Éº?%@G^+¼+tWø7|»ZÅMÊ¦¢&?)LnW? ↩
    Ñ¤ÝÃùÕÇ?w?$^1$Ú:þG?Q0ùj?É52][?¡\ensuremath{\lnot}Áæó$^1$ÓÙ?1ä.?ëÙuýLvXù?S?Ôôj>
¶??Èißíþ=ß?î??*\ensuremath{\lnot}
ìÌç;=½g$\yen$§??®Ü5â¨?\textdegree{}-·cÍ»JEÕ¸tciGemosÍõ}Vl®?ÆAÑdÇH¢Ap¸£¦½®ïÛ$\yen$^ ↩
    º_ÇïkÁ$\div$ü½úßào#sp
???0)Vl?àjZ\ensuremath{\pm}¸½ó;¶‘$\yen$5ÑB\ensuremath{\pm}çú^5iU//?IÔ?&%<Rº~ ↩
    çÆñÆëÝÝ?õ?N»¨ûv¾?oL?T?(H?()rÔ$\div$,X\ensuremath{\pm}Àb¶ED*m½N¯>áÒ’:QZ3Áâl??Ñ? ↩
    kêil@|)3siÂ}»©$\yen$ªÛu´?çhL?<9PdÊP?çÎ¦PBÄwW?-(ºÌöðÅA?ÎÅý?Ã$?Õ¨u?ï^È$\mathrm{\ ↩
    mu}$&6èìNLÑ$\times$k¯¿??·çw$^3$?·¤$\times$W·âõîoñô8¼Sj\?é!Xà#7©\j?Ô?0:w$\ ↩
    times$~o[ôzÞn!ÂÄ·:\\ensuremath{\pm}å!$^2$[cÄM=ý?ÇÉ)Ðâ?4Ý7m»Í4?Eq?Õ?¢?IRP¤6l??g ↩
    {?X\ensuremath{\pm}??$\times$??özÿ§¡z?ÏDsÜåÁÜX‘$^2$?$\div$ööv»¾’2aÆLÀ$^2 ↩
    $ào´Îî$\mathrm{\mu}$ò@y:xF?F\ensuremath{\lnot}1 êQ|èd\ensuremath{\lnot}:·$\ ↩
    times$l??L??òú]?_OÔÄ
Ðê?Ó??Æ¦ÑdMRLý$\div$Ñ[;uâx5 $\mathrm{\mu}$7N3¯&Ô$\div$í]ºã$
?e$\yen$S%iR(
?7à’:Áh½Ú?X.ÊÄ½?$¶hÏZ£´d¯?"Q¸l?Î2!~í?ÚÆï$^1$*wO¶vë?!*_?0(9:$â‘L§NæP?+@TÇÒ?DèKîÖ
?\textdegree{}Ã¸??$\div$?½=!?ÒÅch6ÿ?Ê??B?0ùWSºLÅ?ZMã!IQÛ½©å£ìÑ»ª?¡NeSÅ$\yen$L’n$\ ↩
    yen$?U???B[[Ë¿¿@ÍyòV[j??mªÓTËÊ¤iV0Qu4¡J?{Ã?ñAæñ$\mathrm{\mu}$Ów@f»{/E$J¤Ê
?ò3R.:xèË$^1$JÎ?\textdegree{}$\times$ªÿ1]Ø*Ô?@¼´M\ensuremath{\lnot}øÁ$H©5?‘N=$^2$? ↩
    R

tÓ?ÁÆíÃº?&Mo<xó??w?Ñ2eÑªxRl-?NKPH??´:å4Æöý}x?ÌæÎ.Y‘?Ù\textdegree{}ïmÏÆÕßöTr#É¨v$\ ↩
    div$Ñ?«2PaÅ’^ë$^3$%?yb ?8Pð(
Y1  ôRA7^äkÜ.a?Çg^2?$\mathrm{\mu}$  ?ÍuÞ?ä
AâoÃ$\mathrm{\mu}$Å?©/?\ensuremath{\lnot}$MöÎï??(?\ensuremath{\pm}?©?Yì©ÔáOB}9 ↩
    ª¶KÃ½½?E@Õ?åZf,:Wj?ÜVh¼úÓ?Þ\ÙÀÝ$\mathrm{\mu}$ÁØ\?Ú®¶èÔ)VhNá9ÙÕÿ]Æ\ensuremath{\ ↩
    pm}?½ZÄ$\mathrm{\mu}$:ÂÓ2Js@? /pAzDÝíï’ 6?¯èÍiÛ?æ?ª·d?‘ ?®oíZlð
;y¸§B6ëýÝgÚ¼R?%6¤£?N?ÖjØ B¡9à?~‘?v]$*acz‘ ?h?â-¸Q8Ùùâ?þmJ|;q">:C½çoÛøÚo§*|·?r¦Íw?\ ↩
    ensuremath{\pm}$^2$C$^1$?cÉ?’?9«ÔAóS/
rrN¡5$\yen$ÐL\ensuremath{\pm}ùAàj$^2$dZr"i)3¦?më}ý·{z¯w6ø¦D\textdegree{}áÆÛé_?<ô¦Ö ↩
    [’?s??WOºpoÖà2é)\textdegree{}9?Ô¶Ëïò0y¸6W\Å¨çÁÁ*jð?P\ensuremath{\lnot}%I
```

```
8uô7|ëqäÚî :ÛiÕ2D 6?G?"|4ãRb>??NA1Õ!¯°?ÿ
S?æ?Ði)W?ù?*«Ø??ê¶$\times$ÓÒÌÐÄþ3VèI?<¦$^2$?Ô¢! p ??\ensuremath{\lnot}8&ªhî
$\div$è\ensuremath{\pm}?«è?Hh?Èm$\times$ßÝÚ_1´]~¦ÀØÒc;?[  ¼©Á$\mathrm{\mu}$?w?$\ ↩
    mathrm{\mu}$©£àÑÙÝôÑ_=w û©ðc?ð’?‘øÑÇ?Í}ÄÀd??_®böÖVva]ßyË??"8Íe=m?P?LÉ?äîêÅo??è$ ↩
    ^3$"
%Ô?dÄw?q?ZWÙúÅ|ì?Âç?ç??QÅ9?§Ó
éyË?g4;ëØ?$^1$?$^1$ã¿~bD?<S©Æo\textdegree{}ÖG?ê
âp¿?iÔoeL‘LCã~fÜ
$\yen$É?Y\D½?0Uß«?4"M0?:S/$^2$·¼Ä¿?
sÍ’???ÞMÝÎ-$\div$òhEâåå~évî$^3$¸s.$È?Ê?ÓÅ$^1$R?{{4vøX?,ñ-oXß¡?;t /r~66Öj?*.9%Çë-1 ↩
    _7s.R$^2$%Æ(Aãwmõ_?Btn?@&ïOñYÙp#eè2?ÖÚï?Ü;#?_r1$^2$$^2$+!ÁÃ¤Æ8ÚpÆ?ú3?õàÁÖïo9k?r ↩
    ??&4¢Ã7â@? 0é?&Ú‘ %)\textdegree{}m1?ÜÞ??Áû?IN?(y?U,IV·  \ensuremath{\lnot}J? ↩
    OÖá$?òX?R¡Á¡qC$\div$é*Ç¡2?1+&X4:Ã&¢úv"@P??Dî-?.TY?à®N? ?<nÄ^!$\times$&k°A? ↩
    @¡¤66jB???ëNr?ª£M?Z‘¨hôè«O«ªèI?^ø?ÀQAà+\textdegree{}d¸??BhéV8\ensuremath{\lnot} ↩
    $  P?¦1aè=n922?J|3’S?I?Á©TéÒ 8
??=7yÁ#Sj
DÑN\! 4øÃ??$^2$‘¢;eofu)å*
Ä8$^2$ÔæÇ +sq?ñ§?V?åfN?4q11j~)L©Òòe?=HItëVanf©?È?#?+Úd2?ª$^2$¨bTúÕ’?¢¨É ?QAÆ??8Ô+ ↩
    ÀÑd?Â$\mathrm{\mu}$$\div$???"Æ\textdegree{}4ä&‘´hOl?)?Ês\textdegree{}q&IRZ??îâÆ ↩
    -ÄAâÑ/BÉ#*cé?É?OXÈ*I?Ý?ÁER.$\times$\ù¸?â^_?R¡£?ÑÃs?B[$^1$ò Q$lñrfAMF;ýöß·\ ↩
    ensuremath{\pm}ÂT ??$1rã?D\ensuremath{\pm}??¤¤?Bè?Ê$\yen$\textdegree{}H+[ÎMÍåË ↩
    ??åÔ?Ä?qÈDL5©A?qéF?j£1ç?£VÍ¸Äâ.H?ï¿?¨À|D?G???Ð)?ä¤o#äÞ?bÄ6,¯?$\div$%?+ÌR¨hõZ? ↩
    ¤à¢\$\mathrm{\mu}$$\mathrm{\mu}$ª???Gðm@ÿûl«àì$^1$CN2?$^2$©R?ñË?äÆ2‘?+V44î? ~ ↩
    ¯yw|ÁxWx«&>h?  ¤B$\mathrm{\mu}$¡F¤1iHÁ?ìáeQ¦ç?já¿$^1$¶õËù?ë$\mathrm{\mu}$Ú ↩
    ‘’?:<3X?BlR¶ërYe¡?Ã·´º½$^2$½¸Ç‘¡5?oG?¡à??$\mathrm{\mu}$yOÛDÞyI‘?!A%Cñ¾Ûý½è.f?©! ↩
    §Z?¢?4l?$??R\textdegree{}W"WL?Þ\ensuremath{\pm}æò\ensuremath{\lnot}ïî$^1$)É£@EÐ ↩
    ?m?]Ì??èIÍ.¤jÇT??çÓD?q»?ÆË?rcÕ1ei?¤\tªLm?5¡JÂVºà0ÖÂÌ@?8n$ÔkR¡?0»?ª?|n?N6\ ↩
    ensuremath{\pm}S? Pám#{z4~’Ã?L©q$\yen$LÜá?<CÊR£Üð´89¤!?Âê¼$\times$ì¯Ë¨?h2tÑ$^3 ↩
    $Çe?rg  4ò£?ªh  TdÅ4B5Ñ A(RÅ@iSj,LáS?#4©??&Ò:¾©??Ý©‘®Íó?§Ê rcå~Ô4?4 HQ:(ÑÐÈJoæ ↩
    =*¤½_s]F?Q?È  ???O??b?k?TNa??¯IÇa?%¾eªUÜ«Õ&Õ¢0BÉ?P\ensuremath{\lnot}ª«IR?Kó$\ ↩
    yen$FXû_?
$\times$=?tj?:N¸ÆO?:Áö?? zc»ZÄb??¤BwzÏ?Î©ÛC??J3ª¦.Pá??L¨Hûÿ’9:\°Ä+ÛÛVt"?9?ëP\ ↩
    textdegree{}6PÉ$^2$¦)O.pÍ8?&§?Aÿ.3º&??@Í?Ðÿôq?Då’Ja*?4m?TK?8y_zé$\yen$.?S#?\ ↩
    ensuremath{\lnot}BqcÈ?.‘áÃ??NTa???i¯x+??ÎA?é?yI?[åY¦?ùc?¡Ë8dárX?D"Â?«SFeþqàÔ
DêáC?ác??_õ1?õ?vA?mè’ÁU4Õ{ý$ò£?ÙjUH
??.|Á*N¨lp?M?R&Å~B{DÅ???õÏ?Â?ÎZ?$^1$?S?>©¢$\yen$ÅJLxôÃQ4Æ>}?céB*EÈ@3DÔòD+H¤P"ËÓP( ↩
    ÏäU$’??d???À?* ??f?)r??«V-\)a» %?QX@x!GbxªH,$ÃÔâ©2p?Kªh$^1$D$\yen$(½?Ø|GeÕ69¿,R ↩
    ( Q!c1??zI~X¢HÑãÉ\ensuremath{\lnot}¦lÊ?ÀF$\yen$B$O?xKÑ? L??c8 ?’O¦\ensuremath{\ ↩
    pm}:
jHH?P_*©rï?¡èh¢F-ùÒ¡N¡$\mathrm{\mu}$$\times$Ôe??¦Ð©??bK¦L)~?8ÑK??§?$eê?¤¶Äª¤NTÐ ↩
    +?;=?4$\mathrm{\mu}$??ªé*?©V?IäâDìÂg??%JJT3Ä4?Â¨cÊ??D¤è?Å@?xåD«?\ensuremath{\ ↩
    pm}(¦L?¦=?ãüÆ?P’U|$úT£Q$\yen$D\ensuremath{\pm}*&H?0Å«?-å?B~èQ¦?©N(Aå.LaB5^]£(6? ↩
    é¡  ""ñxÍEFJ)È©6·çVl*?dV?$^1$ë?@!B?y??~S?lLðLIoRatÖ?N-CfÊI‘dý
þ?6Hók?+Tô?KTÍ  ’Æ?$7¦Ï65*4æ  X¡r??? ??Ì?!?m¡êD´ò$IY]dO’’N©??U,DQ? ®’Æ??’Nd¸ð$Åá? ↩
    D$^1$r(R?»$\times$$§!B???\ensuremath{\lnot})#M6
£d©¢ÿÙ?yj)ÿÒBå<$\mathrm{\mu}$©I?®Â¿$\mathrm{\mu}$ÌªòA_?½JT¯?çoÈpæ=s??t\ensuremath ↩
    {\pm}¦%2£?>|$^1$S?Er?Z´ÑÔ¸è?:  ¤Æ\textdegree{}8xñäHÔ?<1£??\?èÉ£???ÿ9/aI?|’ ↩
    ´hIþ?jK\(lñ???á?X
Õ?M$\yen$\ensuremath{\pm},@Ti??H®½yàÙ§Aå&\textdegree{}DR\ensuremath{\pm}*èä?~TWÐ‘@ ↩
    ?\i?gR?Tñ(G®ÈrgM’Xà?2Àú
ói*PQAÈ¯$o<Û?|<Vil?!>?"$\yen$$hteOéÝIðbGFéÔ(N6$ÃH>xÔ)Õ*=t¨$>xÇ ÙÎB?¦O$\yen$Ndõ@? ↩
    Ûr?åyH?Eå*ä?l?g??HAHM$á(Àã\?aBF?YpãSJ5??h?_?HWÅÎBsGÐä$FTª%?ù?%D¡ [??«(Q?0\T?É? ↩
    dÈQ?bS¤ÐNlþþ,ÉQá¼&lè?¨J? AH?4D$^1$?IM‘ñTfÁ
d(¦Md¼?û¯wAnAÕÖ]B}BBÌLÔ?2ö:&ÛÖÓ
7tÓM?j???7Õ?ÜÈI¤fem¶Ûm$^3$#r9$$$^2$;$«e¶[gÇ?6ì½Émí¿[m$^1$[m¶IgzärG?ý6Ûm¶©¤HI
?????i¦ýKß½Â$\div$?ß½ëÝ¶$^2$6Ú$^3$^ª?Øªiè$?Û3e?$û?¨Ô<jCBÒd???ÜÃ«¶Pa?+R1
¦XRö¿eêÉ?TM??á¦?«Ù$^2$(?q5cÔ§F©©?\ensuremath{\pm}ZY$\yen$I§R¤Ê®?VB
"KB& ?àUl!?¢?, ÌÙ1eÿãQ?}~raSW¦hUJJd
?PÃA"?wÜé´ìÏ"chP8óÒ?z"
8+©T2‘N$\yen$*$\yen$m??RBu\ensuremath{\lnot}¾Å$^3$?lÜ
ö?$^2$H?ý?-èÚÇ&JyÚ$^2$¨ã?$^3$LãçÆ¯Z$^1$3Ï«L?t
?Ñ?TET@ÉÅSl~d~T?B\textdegree{}$\yen$y
1¶0UÄÚ$^2$‘íP6ÁöR ?@ø©l
?¶-?TÃjÓ$\yen$R$\yen$? d ?$\yen$D?Ý¤?P?\textdegree{}BT???Wâ(PA
?ç_aàTÀ?Õ?ì?p£$\ensuremath{\pm}Y?ÒÙ<+!mY?¾*! f ?X>$\mathrm{\mu}$SBI\ôR??]ÈÑ"dãe*u‘$ ↩
    \div$?5*?PÅbD¤´????BDSé¤H?èZJ)?¯õþ??N©¶jS\ensuremath{\lnot}C\ensuremath{\lnot} ↩
    s¤?A&?X%z
??¡_ØL$\yen$bñ?VÄ@ÅN?ÕR
0%d+A?¤FÅ6A?k??M©]?©u~ÍA-£ìZ??j4Àâ’L"?2\ensuremath{\lnot}PnÄé"Ù:D¢?ö?D$Â§Id?õ£,#dà ↩
    ??}M¤?&qO$^3$d?O¢%«@Ü?ÌAGL?6øÁA?$\yen$JE\textdegree{}a?\ensuremath{\lnot}dÁ&R0b
¡@B? u!Y<?rÍ,Ö6Å?¯‘?s??M?åe%VQ??BÐ¡¤?daí??RLBd«R?Dä©L §??ä0’??¢? ?N4
?}jç!dÓB$^1$õÏ$^2$("?AÓ?;??½L?v??tC?Nåb?ö?2¢?L??*PÁ®A$BÐ;»??qÖ,
;‘?´
®Å~/¤A»R"&å?‘
aÒ¦?HF*??5
EJ@0ÔÃ c??*A?@\textdegree{}À.$:Îæ
$\times$gsZ?Ii?<??Â??ABH(0~o??Ñ?2T©@©$\div$5Ð©B?é?$?º\ensuremath{\pm}À¢ÁÃÎ?´
!?@?Z?:ÜÃ?(@¿$^1$(!$^2$ ?ÌA??!EL  *¦?à?$?ëM‘Àí?sç~aÜ(Z?È!$E?Å‘?FçS$^1$1?C"I
?p‘Ó‘Æ???ÖÕÈ?ÿ$^1$n,óí??´??åE’ûa:¿Ú? À$^1$R?¾? IRîaàÊ?þ/þaÈ}zJÐ7¾ç~| ÓÖ¿ï?ÿ$\div$? ↩
    úöh7__]Înb¡ëø?$\div$¤úÐÝÎþÖæ
þüÑÿ7ÿÜ?sR¾%ÿçÿýóýéûâ?_ÿÿø?ÿy¸Ä?j?uÜàÀ$\div$¦?Ý??lÓ8!p$\yen$B*U*VR½Ï?x?Þ¤?Ø?8? ↩
    ÔMÅ??‘?’W?¦?"5L??$\mathrm{\mu}$«$\times$Üi4,‘e
  ÁnD?I*ÉÜâÕ«X$^1$p<WnxÔ$\mathrm{\mu}$rËeZ
??01?\ensuremath{\pm}\Ê$\mathrm{\mu}$iP:lÉs%J.\$^1$&Ï¡FJÕ?O~FP©I$\ensuremath{\lnot ↩
    }??ì4jÕW?>Ë6(«r¶lÕ»w
1^?-%Q
??ðkqÞëV2JtiO:tÙ$^2$å?"Üþ?
sènd Ù§X8?à?/@!PS\~[Û¯^Ñöå[ñw?ð?¾zöòòÿ?»Û»»Ç·O??Ý?~?P<ãL£&L?¶âÃ$^3$?$@?G?&Ô‘??D©t ↩
    (?gT4Ð$?Ey?¯’¸Õ{Ø‘(p#5D,??wtòâð»Ü?,??»ü,?ð?guwpá»fkT$?Ù?QsæL?cb?(:zzZz:p‘ëkB?.Ü ↩
    ?ÁTªi~??ã??{&P?É©T:kI\$^1$zöM..Ý»utYxXwø7øxýY‘¿úêà¦@Ud©
?G?VÐ_’?ê$\times$ÖÔÒÔÐÏÑÔÓÑÒÓ\ensuremath{\pm}$^1$$^1$cH>ú4«?  £d??§?ìAªP¡H*GV$\ ↩
    mathrm{\mu}$k&
¼º¸pòöè½Ö?  xX»\ensuremath{\lnot}-Ö&.?w$\times$···n[3b½q# V§Jt¨àFÐ$\mathrm{\mu} ↩
    $õ53ÀX?~6?mpãE&Tùû´??KhPF?é!Z\ensuremath{\pm}«f¼îÞ8nòúêòÿ??»üL\L ??ñãÄlJ\ ↩
    textdegree{}GæÕÌ$\yen$6VÜH»ÓSSOq?jAÕ$\times$
$@*À3í)æR5?¨ÚR8¢«TJ\ensuremath{\pm}ªjÝ{kp0MÝåÐ1P??J¯ð\textdegree{}Ë¼zíÍ$^1$M¯\\ ↩
    ensuremath{\lnot}Q@$\mathrm{\mu}$«Õ"<?ö579¢?U£$\yen$«¯?ccb??.LùÆP¢r"T?
0©É??Ä.jÑÓF  ???xYx{¼\|kòï\textdegree{}\ensuremath{\pm}\textdegree{}ï»)ÁE7d¨tÉDE? ↩
    çÉ#nDHZ:\textdegree{}u4´@¾alìëêìG¶e£L1??À>Ñ+Ó?¢??tÕ6îÊ{x?y\textdegree{}ß> ↩
    Ýuyº Ýqn?l«v*È?+IF\ensuremath{\pm}H?ÀH"?®~®ÆÆÌ=hpâFf\ensuremath{\lnot}Ê?¨\ ↩
    ensuremath{\pm}R$\yen$$^2$?á?dV8vJ$\yen$\ensuremath{\lnot}ÜÜ½x
u{|[íö.
ÓÀd7Î-Ý<våÀ ?:T??êÒ¡È:D?Lí?@5ï;OCGO[N?}©raÎu??% :6?´-?4»bM[Üb?UÉw·$\yen$ãp1ð?È¸Ü
à?¨réË$^2$Z$\yen$?~~$^3$6¶¦Ì$^3$©$\yen$¨?+¡£$\yen$?kj?Í$\mathrm{\mu}$jBÅJÃ?hév?Pê\ ↩
    textdegree{}jUñwOK¾ywÄááq{¼ß¾¼$^1$?º??*6VÐ,uF=©seÃ??ÒÒÑ?ÄÎðeêjÃ£;3Fm*v? ↩
    IR½òç2ÃeQ?mZ8¾ºvêð»§x|mñonìß?à:k}[{
?§?5*?EJ[puu·?W?/ÃÛíwü2(ÍêiÔ¨yòÁZÄC[½Ti?]r]ÍÍËÇ¶åðxØ~I|’ûË¼,?o/?2V?b%H?6VMxPt$^3 ↩
    $´ôt3s»Ý¼®ôÍy\textdegree{}uÑ0ê?lÞ$^2$¤?C?x[Ûð.\îïñ1K½ÃÆ{{ô~ rïÉ{I#N??HYýîöÁ$\ ↩
    times$ÌÑ??u)}ê?ô¦âÄáÕ«^æô¦$\div$W$\div$ÎÞñ$^1$Xwßð8+û¼mÕõÄËûöà$^1$?$\mathrm{\mu ↩
    }$xñ©R?$?çkK_;·ÜðféÝ{¡wöY^ÓÓHvå$\mathrm{\mu}$$\ensuremath{\lnot}?$\yen$; ↩
    âòÀÑwkÅåaÜàâ‘ÝÝá‘$^1$^ !)X~ð_}F\HÐóûÝÍ8>?çk+ÝÏ?c$\mathrm{\mu}$£?åAH4&þý«?e¸)$\ ↩
    div$/ÆÄ¾srÿ!ýÝö>êåÑxå´%@*ðj=-}dGÛÓîös~êfö»=\ensuremath{\lnot}®þ¤)/v]Àñ? ↩
    bÓÞ5szõíÓîNó¾ÁÅÄßðø<L7n]o7e0Y$^1$?¨*>´?$^3$$èeus´´;¿Oc$^1$ÛÎ$\times$éõ¶ðÓ$\ ↩
    div$ç-?pJ[ö^a‘]>ó7Û$\div$Àp?#½ÁÁ$^1$w¿Äl\ensuremath{\lnot}j?? &í?Ï={=ÿ$«ßÊìv:$ ↩
    ^1$]Ì®Ç_QÌêf(¦Êø¦íÊ¿.ïûwÀ»ÄßnËh_
õz´Á@Õ%FØØÔØÔÙÐët$^2$$^3$$^3$òúÙY]¯íÖîh]Í\ensuremath{\pm}?0®p?¶·(¼?ß?Æá|ûÄßpî]7¿â? ↩
    Ý1õ?Z\ensuremath{\pm}àøtó<:»Z?wäîçéåõ:^ÿKã$^2$$\div$R,ùô\ensuremath{\lnot}]ã? ↩
    U7wXwÑó|\ensuremath{\lnot}2ð<¯WÅô$\div$Í4ßnîÀmüÚ&tï?s«ÙÓÒÜèô~>Ï7©ý¾?z[?)‘\ ↩
    textdegree{}Âì¦è//ò<+#¾Æñ$^1$|O½Ã¼tç{}nÁHu
&T?¿ÑàÎ$\times$ør»?}¯ëÐø¸Ê\ensuremath{\pm}.jÝáNn17O|¸ÿÈññ_o|þWß-¨·?ð[#JQ¤DÙÜéüù°[ ↩
    _?G§$\times$ÐéOO?F‘?C<½ó§nË}°æs9>F5ÿ?Éñz?~^?Ðó5 Ï?G//$\mathrm{\mu}$Ö$\div$¾~ ↩
    ÎÆgÃÓée{ÿÇß$^1$aß?oX®KíÕë§Ø?oñó95Åûþ_’ïx$\times$"TetM}?Î-lÏ?WSüÿë?[$^3$ÖþÝ?oõc ↩
    ?&É>3çï®w¯^yù??&éÝûÿ+’"èí.Z-$\yen$$\yen$àëeçgkhge{¾$\div$Ñ?G£û9ÐqæÔj?ÌL_?{Êãæâû ↩
    ?{òÊ»ãz?6Qv?L¼/?c¯$\times$ë}?Û0ûß?W»?fWîè úªU*6å¿Äãp9\+$\times$þèroüKë·ükutëX«$ ↩
    \mathrm{\mu}$Öù~^þg[©õCÒë$\div$u§eû\textdegree{}LÈe?JÌ$^1$Èã¾ñ\¿õ~ï¢þõ}$^1$n/ß/ ↩
    ßÍÙJvWÄÐÊÌìw4¤IáÍ3ßÿ?¼w;2@¾<?¶ÿ?öô®73ÐÈ~ñZû$\mathrm{\mu}$7lã$\div${?\textdegree ↩
    {}çÐÐù°?G{:?áí?ÛP¾á?ð65t)äs??ÿ?ô»$\yen$oý?Vñ}V%*ÁÅ"’_»«%??ý;J??
*?Ïs£ÓÔÈà?OGo?îú?|2\?ëÉÉò·ëen=$\div$?Å?ÎÇ?*gküÝ~?ô:ö?$\yen$ëö}ßÃøwç£$\mathrm{\mu} ↩
    $êz?âÖ3ò}o?ª?Æ6.ß7àêeh?5m®gRhÀ??8Ó~O¡:?øD$\div$:Ð$\div$ÿwÍó_?,Foý_¸ý?ÚV»Þ®©óöÏ? ↩
    j???KJ¸°?ëèsü¼ådos¤Ãér½<%§:å~OÇ°¨?Ø7¤RÎù~?ê+$\yen$?ÃR$\div$NIï~îr¨?¶Ê~ ↩
    OjíþG¯Ëpfi~?8ìÉñ[©$~Ï¿ÓÒ®½tí¨óÙßaàÜÍêt9û^·?»|ÿn?Íçø¾çÎ\»6O®TÙúýÉ®?$©©Óø°?#t] ↩
    ÅÜl¯íÏ\ensuremath{\pm}ö)Ü?$\times$En?Gßæo.§X¾Þää??ZYéU?(}oç$\times$?¸+$^1$! ↩
    c»âéx?óùð?ÑO¿ô?ÇÈõ¼\ensuremath{\lnot}G???Åû$\times$WN?{u«?$^3$ÕÍÎé£w
o0®Çíý?©~E?¿\ÕMù_{À¶AÞ’æ\ensuremath{\lnot}J~Æd©[u(êw2$^3$ÒÝá?[ÂîJ?$^1$£ú½$^2 ↩
    $V¨O»ÙRÃÇÉÈó1Bg°ò<â\ensuremath{\pm}?\ensuremath{\pm}©;;2
  z~ûëgÜ^F9e°?$^3$ûùÈCTl¿¸/ò}\u·Ûï6æ´Í©Ýìö¶§R\ensuremath{\pm}@®Cÿ7#]éHfüËúÔÒË¨’ ↩
      ñ®_#î}ø? xë°ßaØàèû}.¿c \textdegree{}x

æFNO¯êâ7F?W¡üùüÝ?ýý?¶¡ò>Ü?¯?iL~$\times$¿î{^ÏK½?]Ùx8o$^2$‘~O[î?~ÇÇü½¾o7ÙÚ9ÆJ?ï\ ↩
    textdegree{}çÑ$\div$þïþ?o·Íös¤uåzÐ2r2‘s?ï?mt¿·í{>Ï$^3$ÍýWx8p .ü?_a$\div$?è~P| ↩
    ¿§Ýé|?Ç¼¿?ÎÜù=>G??MH}åís½ÑÍ?eø·:«íÏ¯ÌhÈþ<?’èÊ!$^3$¯¿Þªå1\ensuremath{\pm};$^1$ï{ ↩
    ÿ?Áý¿ÇÏÔþ?·Úæï7>\>¦Oü$^1$°Ú¿Ëìû?«*#?7Èö=c$\times$ûù
¡etú=?Ë$\mathrm{\mu}$.Ö¿Åíó?çbÒ?ë$\times$x?éhC$\div$ùý?XÁ·ù<½ëÿ7Ç}~Ò/_©Ûíu»1ZK{?$ ↩
    ^1$ÞßKs\textdegree{}üÙ$\times$MZn=?¯Cn‘D>ÈÃw°Â,¦ ?ÒÍðÇ©"‘1f?>$\div$·ð~/ ↩
    WÐëÌðÞÜà¿$^1$~p$^1$ß6¤$\mathrm{\mu}$®øE$^2$rô$^2$;W?K>x?_8a??ì\ÏÙÎöü?0ü?Y? ↩
    F4èïvTh4Ýqß?Mï?4hLnïSFx;? U|Ü¿ÓÍ}??Î?bðN¯óÛ$\yen$bïÅ»l\ensuremath{\pm}»?Ë? ↩
    ÌðàiM¡U  8WZ_ú$\div$yÿGÛä~??4a‘ò/hÔ’æù$^3$ös6?Æâqo ?ÂõE?/\ensuremath{\pm}?QÑ? ↩
    Jòû_ÏÉ?.aªD$^2$¯q,Uy§ð|ÐûÙê_rü½óqÅAI5$^3${ûrMLÅÍÁ"RÊÿÇ4Ç»Ø?=hk¤|ýáå§éü}~ÞUKÞ? ↩
    C$\div$uØ4N$\mathrm{\mu}$Ó
aëÉ <ä??,öÿ?ô?#þõÖS&[´fïõú½Ø9$^1$JËór|ç#2?â ?OOf@?Ò@??$^3$^oíç%cÇ?Îa~çïl$^3$¾_?,©
ù_s’?H?WÙo;/Á¡?¡ª?]·d´|ßotP?s} N/2*\ensuremath{\lnot}àÓ$\div$úz9]ªX?Rñ:¢?´½L\ ↩
    íxs§UA$^3$Éoï~Uö? |$\times$~ñÃyï5?§Õ$\times$ÕÊ$\times$kädÀÄFp?Ü%$\yen$Ml½yö  ↩
    uÉLOÓö¾?^\ensuremath{\pm}Ú]"?¿#"#«Ú?ÆOªsï"?-eqNu$\mathrm{\mu}$"O8Û+Ü3?%/ÕÎ.?~ ↩
    þsÎÂFË??¼?\îçOUß??N?A??ÉNÚ¶uZr%t(^?X?Æ??é6?<2Zîh$\div$r¼?1ýLÈÕ?b§??C>i-?pçÿÏ: ↩
    ¯GñÀk~ÂF>TDJ¿ØîfüýLâ·ð9TpZÆÕQÒ§J?XõKB8\¿ÍÐ_2??¸?*ËÂ4v¤õþ«P¯#%ùC?SU?#fÉÒ¶õ!"Ð?/ ↩
    ·ûå¼?Ñ?Û6øu  lãg7$^1$ÝïZnýÿ?eÕ.R2SiÒ3gfÂªD(ÿ§öçÖòý?8Î?Æ$$^3$?ö?$\div$w$^3$)î?õ ↩
    <ñ’V?hP6$Y?ÑM_$^3$ÿðÀó?\textdegree{}"z?^ï?úþOÑîÿç¨£y½ä]$\yen$Spùâ$\times$#>? ↩
    ¶¼É$\mathrm{\mu}$©TT1ò½«:>7??$\mathrm{\mu}$¢+Â·N?;6$ìjH
à+? á‘ÜÌ§R]?»s©"2ÂýâW§?þ?òa]Ó§ÈÚÊÎDuXªÙ2‘Á¡+C¯???Ó?ìöäFýý~Õ~<?_?Ñiï?ÇtRÚ\ ↩
    textdegree{}ô´´aTL½eìIR$^1$[V-?dÈÓ)S«*lê¢?óë?õüþV??e~?ý«FséC?m"f+TH/fT©5« ?¼?6 ↩
    Ê¢lñÝøó$\yen$aý·ü-öô\ensuremath{\lnot}?-Ûf®?¢eI1hZ9? !\??n?tÝñ?:¤È\ensuremath{\ ↩
    pm}¡lìÏWäÚÅò8{ç$\times$#?%¸¯[«T$©Ø?6ãÑ¢xË?[4h@ÈU?Ë(P6?JÍ.ÞÝ6Z$\yen$¡X6åÜ=)»?à? ↩
    TaP >Tê?jà\textdegree{}?1ÕêS6$\mathrm{\mu}$[UiHKlä?Gy-½n?8ozS·.Z3\¼qB\ ↩
    textdegree{}a¶NBÒF$^2$ ?2\ensuremath{\pm}BigØD;?iÓ5?ê?«Z\ensuremath{\lnot}?i- ↩
    XÌ§-Ê)«&KI?I#0%PêD?; h?G aë‘îûÌHÕ+EÂ¢"%$\mathrm{\mu}$e?B$)JÍPêÀ[ÐChêue§ð%$äe? ↩
    ¨ªe?¢D0É?å’8(¤E$?:Y£\?ö’DtááR©E?è)á|äÆ8¨ÐATÖ^pJ3Þ:i9õÙTVe($\mathrm{\mu} ↩
    $EÀlÅ·6Üe¢A?v#/Z\textdegree{}kÃ$^2$RÕVê-É°Lôn?v¡5ÙäÞ?Üt[?©Ùf=å"_èn, ai jb?HSb??X ↩
    ?¤¢?B ZM
Ç\!$\div$?%T$^1$y+ , ö$\mathrm{\mu}$[?È$è°Dk1TæuE‘1Ö¡TåÕS?)?$\times$J@Æh?f0@?óT& ↩
    «m$\yen$H$\times$FÒYL Ná"cÎUU???i^Â$^2$°Bq(’~-tO0?>SÒ?©:??i8uíX:\textdegree{} ↩
    içW?ÓÕL?ê«d")1B3Ñ[ P$\yen$?M-1f\textdegree{}ÛVÔÈ8Ø½MÚ$\div$ÛDÒo9´G:Î§&=2)ç-9) ↩
```

```
É5½v4Ï?8ì??Âh¦¯&¡5RPAT$'?Ií¢0!?â?Y,64jäÎq¶^+'ÜÐu¤¼uc?J¦ÿU½£KÕ½DÀb1íÏ¨úS?ãÔuf»D ↩
    ?/s·ÚìÍ·æÔ"ì§q$ªX\textdegree{}Q$^1$îz_'Þ_u){?MuQ@íN?+;?Q?ÇiUMÎ©$^1$QQ$\ ↩
  yen$dJ¡?$^3$G¡{®ûiáê®^Þ?ÂÛi?cFjÜÑÖÞAº?Ê0mäª%='§"Á5?Úëmá?=~Ù??ÄUfÓDVP®?L´åf¸ ?$ ↩
  ^3$Ü?ð? *ìé¶$^2$½r
ê??^Dâå[?$^1$U´ö+é=~kÊéÅ¤ääç??éì\ensuremath{\pm}ÚÆ?½YI?~~$\yen$Z\textdegree{} ↩
  Kad@Lf¾©;O¶XªJVs\S°2Tô?ècø1%ª5?yÇ\textdegree{}LÏP1\ensuremath{\lnot}V$õª´? ↩
  CZRIÓ?5*¸\ensuremath{\pm}Çd$^1$1h8Ð·æXÃM
ZØ;'2J¦V"/d¨À%ªÉR8Twt?sU=c[F$#?ÑP¼Ë# :cÌ}\ensuremath{\lnot}¤JRkI?´?6Òé«Î´MFhK½>¸? ↩
  o$\mathrm{\mu}$ªÞJÈøyÏi?$^2$$^2$$Q'¨Æ,è~òs01KK#«8%QsQB??? ;d?T)£a!?hPVÆ$^2$´æ/ ↩
  Vës[)ÉNy$ÆyIHÍê$^2$¨A(M?$^2$¯©*=\?þüRèÙ]2öç4\textdegree{}IÍPy)m¦8ä Ú£yã-< ↩
  usÚÐféÇ^$^3$T??ç5v5uT©9bd?Z\ensuremath{\lnot}K?ãPb¡m$Ñ
uvSÂ!\¼]F¡óÚí¼$\yen$®Î¢$^3$/$¢©?¶
»Ör)ÊsD«G2ÕÿO¦?a5&(¯Ö©U?%?¤¼#d¿?5ËÐZ«\ensuremath{\lnot}?$^2$Eõ2éIW+nê\ensuremath{\ ↩
  pm}2«=e?4Bl?¸?}$\yen$?r\XPæt?gheñ$\times$?£Ä¡u¶¼Ç?kk1),$^2$ÒJ$(¦_j$^3$\ ↩
  textdegree{}G9èÅÎoE5c%>Èè??'°>Òz¶ER=Gr?Z´DæÕ?_JâBôÖ5®¤ñ?1zS
??Î~!e5½?f¼?Ä#_?$\div$5\ensuremath{\lnot}???\ensuremath{\pm}Ü?Ä¤&]%%¡$\yen$c&û:àBg ↩
  -SH? ¤^ÅiG$^3$$\times$¤c
~Ý?Ä$\times$^hKC9Õ2gÛ´7?$\times$T»Ú·6$^1$T1uÊE?-$ËL 0#P$\mathrm{\mu} ↩
  $5WÌº¢¸ôÌå·¨ºgR1!4ªÚ3???}5) 7¤©;ã?aBÒË",Y5Ý?? !U\ensuremath{\lnot}ªJehÛZ?4? ↩
  mìÖñÊ~TæÓÖÍ#???Æ>.ûÀÒ~¤©D¨ÞAt-öÊRVÍÆÓ»6?1aSV
Â ?H¶uÎ¶/ÔyÄ&9J'¨æ*n*?a©ëÍ]5Äb¶ÜÒR?$\yen$9\ensuremath{\lnot}ö$\times$S{,ac9Å? ↩
  ¦BN¿Õy?ÌvóÔo5ÆÉUÙP?$\times$ÏD@%a)åJR:#$\times$Kw¡{^ÁUP?I?Bd¯?ÁÁ!ðÎSI?5J[% ↩
  ÝB19ÛZ?2gIAÍk-M$\mathrm{\mu}$R£bÃ?Lp(?bôèÐ½Ó.$\mathrm{\mu}$&?XRØÅÖ!|«?Q$#\M$[ ↩
  a$\mathrm{\mu}$°ú®$·J?ºü¯?@UDZKvba?Ps?O(??3\¢î@ùtÛÚ?P?)H~ÂI¦Æ¡#,¦æjÆ<'eEi+}§¼îa ↩
  ?'+?ç:Öõ?oB\textdegree{}'?WP¸Û¸©R 6$^1$9??\textdegree{}Õ´?E5«§¤?PTT^H?rÒÚäÞS$? ↩
  ÄÂ?Íw¶Ü:W~ÎªN5ó?«ªÞÒ  9ÔÉ@H£$?*?iÌ]GF®?>d"?$^3$WmUX ?å?eZi$\div$?ÝGlzbOV~?|? ↩
  úÌÄ\ensuremath{\lnot}?*7®sF¼Ô©VÚ?IS$\mathrm{\mu}$Ö½àÄ[ULI?F??JcQ$\yen$2,?9Ê?^ ↩
  í8\ensuremath{\lnot}\ensuremath{\lnot}CÓ$^3$-«$\mathrm{\mu}$n'ÄÆ(ãÕ?$Á.~Î?&$^2 ↩
  $áj¶ÒúöÕ@©ÒÛ?Zc?ïÔÜÕÑnf6ÞIaaA\ensuremath{\lnot}
?B?>0D\ensuremath{\lnot}ª0Öt?F&:# eyä?´á?ÐV?é$\yen$Z$^2$H¨?®ÒP6PáL$\div$s$\ ↩
  times$¨1UÓo@©kÒ?oI)6?\ensuremath{\pm}ª?7>~?Ï?o-E$\mathrm{\mu}$\textdegree{}Bô ↩
  ??~?á
\ensuremath{\lnot}?à@?qÃ????$\mathrm{\mu}$ÔÕ\ensuremath{\lnot}%?&?Mu(ìqT
??%:$\yen$?+YlJâ?%$?D$^2$?´$Êà4$\mathrm{\mu}$sL?tà?ê?)ùª-p&4ä%V2¢Y3'Äk-lÑÆ?R?¨T#Ã\ ↩
  textdegree{}??$?\ensuremath{\pm}B~¡ÓPeY3PH?Ã?C+?ªjíÃV-ºzÙÃQÉZ¡{?
CH$\mathrm{\mu}$B¢óãªf®\ê$^3$$\mathrm{\mu}$>t©§ª?}¢$\yen$%S?ËY6¸¸º)?wW¯oY-^ ↩
  ®ÚÝuv6Z´*3nDXzYFAV?ÌØñ?d~?
½s?[»¾Ý½rÝÁx[·«?®!ÁD-;T¤Í?ßf?¯7àé#« xº:{rÓ@Ë?X?M®.îw|1=Ó§?Ø<?EÛ{\ensuremath{\lnot} ↩
  Ü\textdegree{}%ejÖ#H¡?^?^©ñ~¾eC4»9Ò?3¤®a|?XX?äp1ý\ensuremath{\lnot}n7stÍq:\ ↩
  textdegree{}ô¼?N¦7$^3$Òèsþu?U?Íù3e?XëºÞ¿àï1øüÁöq?rßá?õÂð§S?d?Küô{^??cäêÏ7Ùu©? ↩
  ¯ËóN?â·^]$\div$?îò¸Ïü?áéx¼NF$\div$]Áw
ÕÈÏÍíç}[]Ïïú½Îg$^1$ÜÊú?;^´$\div$s>>ÚVy>}Ã¼}ç§?ÐåäácÀü<·ïßð&!Bæ§xz¿ßûüYºù_Ãø>  ↩
  ÞËÊüÜ$\times$áR|§áú?ø nâ'|<?Jþ?§Ä-óþCRf?q:}>·ÐíjfoóÑëfV?§$\mathrm{\mu}$ÕÿN_$\ ↩
  yen$þzÅòàzí?cr @òy?»7àÈ¼'??òlýZ  eeûÿóÎþ$^1$|??k$\mathrm{\mu}$$\yen$08¯Êâ
Ì¿Ëþ7ð nÜ^zÐ<Íÿ£ymË~Y®0n+jçx5ì??éèþË?fÆ´mËPú7V?~?$\div$iÞz0=? [ÀûÞ~6$\div$|?ø0Þ^. ↩
  £¡ÞÒ\ensuremath{\lnot}!ò;ÛöõvêO§?n?ô¨Ôùý«?#ãðe~¸<È?%·?ÏÈ¼ââ«Ãá?ë$\yen$?t3$\ ↩
  mathrm{\mu}$\ensuremath{\lnot} d¿?õþý©?$\yen$FLiA+¡ÏyäÇÃÍýhaúÐ0ÓÛåñx¤¼f'1{p$>çz ↩
  ?kU%$\div$ÏÃàZÑÖ@ëû¼öj}î?ï7?"?ò"3}Êß»â^§$^1$Ý?c?CÅîfÃ1,ù??R¨~?©G_£ÚØôE/?_ñ} ↩
  Áö7á¦UÆßÝoø
Ý-¤$\mathrm{\mu}$ÍÂ©ù½ÍZV'GÛégG6Ë?ê??åt+^*lv?èØ<eèÉÙ¾}_??s¦·)\textdegree{}ô$^3$#S¡ ↩
  ?n?ÑÙÙ{?¤¦1?éFýüâß'ßØøÕx=â¢?p\ensuremath{\pm}ï?å'´)[KÊ'&Ï½??iËóghÔZ´¦+*??«iô½ª? ↩
  ìâ?§4\ensuremath{\lnot}OK  g\?87#ÅsvÔs??ñ?$^3$<?ñ¡Æ+·Û<v%$^1$b¤"?\Ï7õó^ ↩
  ¶WàPßãàºk¾»/?Ñ}utAÕÉXJMçcCf?Ä½¼)f9¸À-?Ø«UÞöÿqd«w[+¤;??ùÓÿ
ø¾'ªùÛ$\mathrm{\mu}$U?dægeu$^3$¡kxileíV]q¿ÁÁx*Ø?ôókEá:¿ÒB?ù?O??ûü?\textdegree{}
?ZÚ´&R0È?GÍàÎÎ+f\ensuremath{\pm}Åaï¸Ûûû$\mathrm{\mu}$d\ensuremath{\lnot}=Bóùz
5R5?£Ñ¨Oëþ?ü?SÍ¼½ô½$\div$¡$\yen$?fW$\div$øºý¼Üû&ÚXóíq¼wÿk?iÑ?·wós~C»7Ëúi??\à}Ø?7 ↩
  ì¯ò~Ýá63º~çC©¯«òtºÙ0g:-$\div$!ÿå¾Â¼À%?=®ÏËñ{\ïÌ6?$\div$ùº
```

```
?úÅcüýûMßôó}s:Fgv=[?ÂÉÉü~w’è\ensuremath{\lnot}é|?$\div$ÿ?ÿÄ?óo[@ü<\ò¸»Ï$\ ↩
    yen$ëßPÊðóÿ<ïgÜü$^1$«Û?iç$\yen$?ç??û·?DRGkúþÞÅðe~àt?ü??
ü~?àÖ¾%ÆG$\div$êTÝþÝ?$^3$úû?yy?8î?ûçÀÉÁ?f½£\ensuremath{\pm}«?\ensuremath{\pm}Òù$^2 ↩
    $¾xÿÜ?
ä~0v{øN?T?2äù_ÃôY3[++ßþ½Mbÿâ?#ÄÅ$^1$½ÆñùK"v$^2$¾^Z?=½]Mxß/$^1$ú$^1$¼x?0=N?eV7\ ↩
    ensuremath{\lnot}??Í\ensuremath{\pm}$\times$$\div$¿WÄQbx{úÉ?>8.ÉNÁ¶
þïk+»(!?¡JMm¾Só{ß?#ÍÄ?óz?3·Ó$\div$v´E
Ñv4NÃô<w$\yen$$^1$&$\mathrm{\mu}$ÌJ©??¾l#«L}? ü?~oëz\°Z¤Ûrk¢Ðêô´aeJYÐ$^3 ↩
    $f6âñÅÂgj1Y!:\ensuremath{\lnot}94
?$\mathrm{\mu}$Ì£$\mathrm{\mu}$ß?Öx{ÜÉï9?XY?ñÁ¸o$\mathrm{\mu}$Ûø:\textdegree{}?¡$\ ↩
    yen$?ôM¼äðqß= ?DÚ3õ<1+(\I?Ñ£Õþóx/ý_K"?:?ù02o®ÌÖøz9\ÙPtôv?ñxøû¯?\ensuremath{\pm} ↩
    ûZÅ2IÈO«%?
É^ÎWêæÁó=<m?§\ó xîÖðu¡P‘©¡wuØcqw»§$\yen$!?NEªô?$\mathrm{\mu}$NFÄÝ:X?jZ9]?ùæÁÝùüÎ& ↩
    $\yen$D.ý/_!åÂ???fA*ª <Í$\yen$Ï¸[çÅ½
&k?RA)pä?Ý \ensuremath{\pm} $\times$ú9??Êõù0§Î+?Õ,\ensuremath{\pm}$ûÿçBª&Þ·!  ↩
    PQÝðøÜó\ensuremath{\pm}¶$\mathrm{\mu}$aN¯2\ensuremath{\pm}õ¦Æ~’Kí\ensuremath{\ ↩
    lnot}vgÉñ{^ñ{¾VD?$\mathrm{\mu}$Ì?$\div$2???$\yen$®$\div$C\ensuremath{\pm}\ ↩
    ensuremath{\pm}j’d[b5Þ$\div$Çãà7\ensuremath{\pm}??Js% !ÏNÔ¦¤««îs¡¾¿e4ìù?97ï? ↩
    «îæF\ensuremath{\pm}QËTÁ?¯S?~ÿ¾xÆ¤-myóf?u?»t9rÁ=jÎéæÓÄ¼òò}?¦Ù)ÿÞÉ¿À&wÃîfH$^3$QË ↩
    ?Ï$\times$Eó$\div$ün?ÒeÃÙÚ$^3$¶mYq#ÒRÝ»6?Éóö?g8Ü?ÜO¿çM$\yen$YÇ?Øô¯ï?ÓèôsêYkRjSÚ ↩
    ?yÅãïøOUIÚÙ2$zdmC®E½Ëuögkø~OåÏÏÝ?är}S’ÌânÜ$\div$ô¤&$\yen$Á5èD^ÿÂãÆ¤m¨ó$\yen$K¤h ↩
    ;Ù/pÜëIô~jbËò98·ø$^1$9ßGWTÄ?°%?óõ?Ýñðq**Tú?äÒ©·?=QØ\ensuremath{\pm}^¤ùqvrû? ↩
    £zÛÇüW?Ã7??t®ÇO2?6+®XÃ!Âáãà7?m*&?dzö¡G0?~Z1ãìëw=¾k2\textdegree{}xÿ~þB??ÆæcÝ\ ↩
    ensuremath{\pm}<?uV2Ò$\mathrm{\mu}$AK?+Ýîÿ?ü¢?9?N:u övè$ ÇF<j?ó¾?Ë´Sßç?bÕ¾?§ë> ↩
    Æf·W/Zb?X|$^1$Ù?¿ÄÝ^/?°@:°R"mF0 Áj$êò’jhÿ?rfwO½\z-»ÇÉôw$\times$EðøaÒ?ä«\ ↩
    ensuremath{\lnot}Ì.{??ö6?$\yen$©ÐFÍDiÒ:?9?m?T?~J?9c?Öã?XØßoÒáã]?¯£¯?yë?¨©$^3 ↩
    $´çu»ÝÞ$^3$??(ÐJ}«46v!í]:"Îdi?úÞß@rmüÀ] d ^¾õ2>ÕõÊ]]Hs($\yen$b;z?¤¯¾Ýî\ ↩
    textdegree{}Þ¯J Á?!!3kZ$K?Ú>\ensuremath{\pm}$^3$?~NõçË?ã?òdY¦1w?O¤ÿy}}2?¤©Õ‘Æ¦§ ↩
    %&/w[$^2$ÊT?Á I?*?9l~|ésgOj7Oów?{?eEøw8ßúñò¸ÑfÈL©$\yen$¨ypdÞòöùÚôÃ?ª1YZF¦¼J?ëØ ↩
    :,skÔîóç~¨¨?s?>g¿ô<ÌRß
Sf|ª?ªmZ?Ó$^3$?ÅNË¸jJ¤H?8\l¨\ensuremath{\pm}%Ó«^¦Þ¾Ì8\textdegree{}åK8_ëü"?-¼?¨[ ↩
    Mßâ½Ýï
>Xòª-¶B :iÀ&\ensuremath{\pm}$^3$"
å¤V»¾v$^2$Ä$^1$Té?J<y1áÅ=)òýÏå?pÏ~??ëvjø>_#?ÿ?~êÖa7M«Ä$\times$ÐÕQZ¡2?©ÍýÓ-?VÕ
1?nT(Zqd?ÎÏSXÂ¼gÉØQ8??[î/~Ý{?&?nÜ0"z\ensuremath{\pm}V?Ï@[f3 q?9??‘Êô&H"$^3$ö:çkÙ? ↩
    ëÐÄPÜw?o]?¼¿/ì¼‘2$\times$%·l½ElBJ?¯7nM#N*cÑ??? ?ãh’Æ)?¯FÄm?=¤???XÞéÐ??Z½Ý$^1$M ↩
    ?9
?@1é§??åUFx$}+?%Dj¤Ó¢$^2$?¸¡Ó$\mathrm{\mu}$fP¤S?I?D£?á?dÙc?kbU\ensuremath{\lnot} ↩
    XRæU? XZ¡Å!1ë§*¢=tQIIe$£IP;%ðîò$\times$BÂÍv
Q4v3Re½???¸ÁBûedª¶&Ùceå7fÊÕKT6TÅ[?~?6ðÛ?Ì?£ã°-ëÃ¾Ò%
a?"ãæÞ$\div$Ù ¼âE¨‘Ô??a?BÁÖ$^1$¢:RÄ¡Ö5T\textdegree{}?\ensuremath{\pm}?aå¢Y¡wQÁëª
â.?f!*$^2$xJ?
4b´J??$\div$éa_W?ª\Ûß¦ú¦3? #$\times$?h©?‘ ìê8UBþAB´fiIiG?[5?,?:u?oMªÌ?¤$\mathrm{\ ↩
    mu}$¦£dQ¶éIDAè?$zûÞ©[($Z§?õ0Û&b"V®8ð\ensuremath{\pm}$\mathrm{\mu}$D+\¤?là?@½,H$ ↩
    \div$óJ‘Ö,?W?ÜñX0Z\textdegree{}Æ©9ÅÕ1LÅÕj?0ôèS´-"vWÍ$^3$=ÞeÎT?¢CG^êVÅäûé?·AJL\ ↩
    ensuremath{\lnot}=pkdç\textdegree{}??Ðl@&¾Ñx+sJ<?-½05ãÐ_?ê*\ensuremath{\pm}\ ↩
    ensuremath{\lnot}¢P^?5m[8i?Ê(Á§\G?½g6??Í©<UU¶?~B
?$ör\ensuremath{\pm}ü8û‘¿0ÛÉÍo-â{C6ªÅQ
ÇL=öÛ?kc?/JÄ?Ub?Á.)zë$\times$P°Ößf\ensuremath{\pm}
{?c?bú?VQA‘zøÝkÑ$\mathrm{\mu}$7pÞÌ+lS?\ensuremath{\lnot}¦Å\ensuremath{\lnot}?ôm?\ ↩
    UÐj?A-®~ïL???IÀ?Y?´Úf1lD¨b»ÒÃÕ*Ír§\E*»8$\yen$?{ÖÝ$w:v?97böre2  ~Æq(/aî°òhDç6ê) ↩
    F$^2$¸ZV^?*ÖÄÂÛZ¢vRÁbÃ?91}E )Ä6v§®Æ°  %*‘!=O?QÀ?ÑcIz¤EP°û?®~Ö?$\ ↩
    div$T7®âßì1A@Ðw9T¶<HÄly
ú3J
eÖÞ/Òÿ]½%ôYUqE$\mathrm{\mu}$ÌY=Áf$\mathrm{\mu}$¨0d?cÿC4g\õ\?oØW)¡¼Ò?Ô‘\$^3$EJÊ%V\ ↩
    textdegree{}Óa©2V)$^2$ÚöJVS?O9T?Û+¨,«SwÙßÇu¨\kÉWÙS7??Ì¦Y¿$\times$RK¾¾G»çw^%?dË> ↩
    S$\mathrm{\mu}$4½|=?S?ô$\div$ûK%$^1$Dg?? T$\yen$ Å)VzÛF?¤öeÄb¶ÊÅ??,èÊÌ?|Âó0D?? ↩
    P!îÉrÒªO@w$^1${½¡G$3)?)}&?W&¡öuÜW´pÚ?FË¼ï’âò´IuDªk &®ÿý[¯#ÜÎwCÕÁ13Z¤’?X$^1$k? ↩
    Eb©Ê¤ÉL«q«ÓãÃX:$^2$ïC8æ??ÂBã«ìêW\textdegree{}EÌÖIÆÌÆÙi=$\mathrm{\mu}$+&Eìq)5¦" ↩
    qF]Sø=)dÉw?©k¤;ð>2sÕYÙÞ% è2?UÿKÎÿÓüÕj
?[Úg‘ÂÿÑqn|>$ÖØÂßõ9{?ÏY1]$\mathrm{\mu}$’?Y¨à?©ÓSññ?y©ÞÝBt$?Ûk?ÉÇÌb
FÎ}?î ¦ ‘ºú+¶?õ¯D$^1$ªLì?k$\mathrm{\mu}$F;ÚÓËMÏÇÅÓL?æ+-] kxxúÌS$a$¸¦rû8@?kEÿÃ^Å  ↩
    Üé¼oJ$^1$$\textdegree{}Heå?mäbí-=0iG"uÔôÌ*8«jkÍ¼??V>~
?T(?^ºÙû^
þeåT®lÓfã\ensuremath{\pm}æeÊc*ÓP·8õ´UÎÃFyoúåÐeÌÖà¢ålB¡ãN«s0.Oýo%,\ensuremath{\ ↩
    lnot}\ensuremath{\pm})í?
esÓ6Kãõ^ªÕ/Tº]C*ábpü7t[~
?O]A$\times$???Þ??åÒÀÓ*é\ensuremath{\lnot}\textdegree{}Ò\ensuremath{\pm}+ô1MºÙ?ç’  ↩
    J)Ç¸j¤?Ú&´R{M$\yen$á ÌQeï~4g!ÒZ~âÁ®¨¾’_\textdegree{}À$«ìâ\ensuremath{\pm} ↩
    íóñÄ1º!3Èÿ?Î«¯$^3$ñýïuË?»DËf?I?Ì?d?+¶Ì?p3ïö
iYë?:$\div$7? É\textdegree{}I?XÍ$\mathrm{\mu}$?£D$^2$??Nç?ÆÊâw®Úg\ensuremath{\pm} ↩
    ·Ïq\$\times$!ÐÄØn??#®\¶æÕ¸1tõ$\mathrm{\mu}$$^1$Ôw?GV?¤ÅåI¶»Õn$\times$BìÁS#fó,$ ↩
    ^1$¸:Ó;È(âý*[lÞA{XñÛço»ïû?éâè¯#mÝV?k?7äÇ\ensuremath{\pm}t£mZ$^1$??«ÁÕc3{½?Ñre|? ↩
    y-æÑYø?ÖF^?S<¯ýò5îèGÇÕ,È·‘õ!ÓL½?~$^2$[?çºpk*v?Ö?®?$^3$ý?ðþQ*L$ÕC¤V$\yen$lWn¦
Û»Ìì¾p©^Ê=¼èíl0}´^oô?=¤Ñ?,ªÚ-ÿ{?Õél_ø^J}¢ÖªpÙW$^2$S?vßmVÀì}¼$^2$\ensuremath{\lnot} ↩
    Ïâõ?)ªê0®¦?%R
(=\ensuremath{\lnot}uön?H\ensuremath{\lnot}êBÚ¸®?ÊÇq?ÚÑ?VYm?©mÎ·?¯Âð\B?/þ?fZ?>Ï? ↩
    £½Ì???È?4ÄÏ%2ýè&N»/îõ-$\times$Ô?%?Ê6eîºöÉ$\times$?¡éü.?Ð9Ü&ö+??ë$^3$Ü$\yen$¼ª}$ ↩
    \mathrm{\mu}$;5Eáf?}R¡??iÔx?õ??z{©ø?\ensuremath{\lnot}?8pèâÁahbcÝ/Õ·3Éÿ?s¸©t\:Ã ↩
    ~C-K®¤11îè,l&?Þ¸?Úc?ãnß?þ\ensuremath{\pm}"ÝBoi¾¾?È
?u®îÒÛU0!Û íù?ôP8]Ü=kÙÄxpÆ??Fád»Ý?CÐ??=ý???>ó "?¾óYYTç2$^2$Ô\ensuremath{\pm} ↩
    àiÕG«ZGÞë»mºxU¾å¶ÛnÛnþã$\mathrm{\mu}$m¶¾ãUV¦¾áJªÊ¿
¢¢£ð¢"?? ???ÿ;¿ýÃÞÖFÜi^g¯?Õ.Ã???º?c3ú?®C??Â(sWz=:AíÅ{E¤Â1}y@§ø9
?R6§b?B1âªnº½*GåkU^à¼\ensuremath{\pm}ø? &ÁZüÁ?Jr?»?oÿ´Ûf?¶®Î>R$\ensuremath{\pm}  ↩
    ?£G?ôsSQ’$\ensuremath{\pm}¸¤ÒS(ÌS?xf1qÁG©Nøèefº\ensuremath{\pm}À\ensuremath{\ ↩
    lnot}Ô¶¾$^2$p@ÁsMÛ5
!bq??haÔ??Z?ÈFúm?T??íCM=?:¯PØ$\div$É?
È!"Ò¯Í?IDÃ?G?Ý?/0?ÊÆDù0NÐB¢?taØ???G Î*8\ensuremath{\lnot}qää¶z?ÈG§?Ìo¾gK?¡QÎÖ$^3$X ↩
    :P¢N!?Ëþîp?Ö_ø.Î(L?HMT£$\times$Hª?4 |;ØÐêsD~ $\div$T¦½K/ ÜÍ«½VyÌêtE?*c+xhy¸SÉ?& ↩
    h»?X?uÅ$\times$Q´ê\ensuremath{\lnot}ÏXMÕ?D£?$\yen$ÉÈ$\yen$hñ4B@ñTÈÆf?@øÀsHLA?? ↩
    ÜðPXär¤%¤Ø~&>pDÐmL??nÒa?=?u$¤NÏ?$\times$T41#È£l?9Iå?Tèýu?d?6$\yen$ie]N5Vp¯?+´AR ↩
    ?k(Ø?3y?ÍkÔ-È$\yen$~zpó/ê3ÌLì$\mathrm{\mu}$?þoZØâVÉÖ£tm?´?NNC?e>¦.õ|rrEË~%-+ ↩
    ÄJí$\yen$Ð®?¼§Å$^2$?b!\ensuremath{\pm}ÖÉM ? ç½¡53j6áÚ?þàéuONA?"véC$\times$@?(2 ↩
    ¤6V?‘mÞ2MªWs(tÈWá¨1ý$F,1‘Ü
è ÈI3,=ÐÑè?oÝ?ªOüö[£+}W~, õ ??$AÍr>  é?D¢n«å?L?}HàÂmQ#?*V?
ü«fiöÏv-?Æ\¡0ð@?8ß}ûu$^1$êÛ¦L´ä?£Óâð?Ê«õQÍÍaÙ\Ú~lUYÈ?bÒlÑíÁ$^3$ÿ¿qáã?ñFÐd?ÓãykÈ·U" ↩
    m?!Ã2?Ô?!M}e’+à¡! ?D?’L=9ú2?i?9XZVí??B8?v?Ê®?zý\ensuremath{\lnot}jt9?kQ\ ↩
    textdegree{}a? ªü B&2?¡( Ú¿?<$?N?DA?I®Å6¢0ôè©Å§8¡ÀR?dëÚüz¯þ?{P+1?0øèà; ↩
    ò5ÏnÍÜÄcÜÓN&
F$\times$"«MDÀWdzlª0? ;#å/g:Ç\ensuremath{\pm}ÎÉ0r?q¦\ensuremath{\lnot}g_?
6ÒL|? ?UTx?»»Õ4D\ensuremath{\pm}?©?d!?A ¡p?{?
)$KÇ@ôè$\mathrm{\mu}$QÆºC?=?8ä»cÎË  ?Â?¸N?$^1$§$\yen$=ÅÊ,\D’?sNàÂ?t?ôU?Ö)
B?ÿÊ)-?ã:]¢hÞEéâ5.?ã^Å~j~ò?Zá?Ã?Ç04;?ý
N
ù ÞÏÅ(?dÙô2?¶«s?$^3$Xarª6ÇÅ$^2$ÍE?\qIkV@Q~¦‘É$^2$·Ù¡©j|þÙ4V·(£;y?NÖå&!?Ù¶^?? ↩
    NÉI¨¦@ÅHÎ*?osÃkG0j??Ýæ?I"?§?ÉÈã9\ensuremath{\pm}A${4r¡?ðÃv?òøY£ÞÔè¡^!dM«$^3$Í¢Ø ↩
    ??yÓ-rÂ=?ô#ÒO:W$^2$wlá1Å~Ñíp’Uû?SK?#D\ensuremath{\lnot}àRp  ãÎË£r z?v+s(5Ð?Ù?(% ↩
    P$^1$Bêª3ZÈ¶^¯´º^$\yen$^\ensuremath{\pm}Rè?Ê[ç[+Z¿oq$^2$:4£a’"0f~a?;g?GJLr?>H" ↩
    ãÌ"Jh?$^2$0õ(?¦&iÒ¯"«Ç§??$\div$a?j¡àÁEaØÔYæPl?@Ä¡6LÈãHå?D8ò"O!&páÀÊ2ÙRæ?*?#?
?T9?ÍF?|AI’´DP£?xH(æ]Î"ì"OK
©C$^2$1ìóe¶?\q?éÆ‘üAæÂq3ry½ÐÈ?w Ò¦??ý!?Ù?C2?ca4@©bÓ?P(¡É?vÐrvOLr?Ó?ex]É"?üd}Yý¸GÞ? ↩
    ¤??üæÁàó?{8ÑHÅnqA!Ý?BDù¢eJ??º<?%M¢Ý?:péÀÐ0‘áb2C?#(icK(?òK?£ù3K:?èMô?GÅ?BTrc??  ↩
    h0R?L©(£ñA|á?$\yen$O?4e ¤??>}?ãÇ @b|ñp$\div$E?4?-H?<1QÒg¤Ë??)-s???TqÇíÃ*? ↩
    Ûe¡Ç2XQöË?\È"Øb$?nE?@l~fÊ8tÒ?§Î£9>?)óÈ?g<tÑÀ3rã?P¢?$\yen$?2Tñ¡Ì?lmb?74X3Ò¶?"tÐ? ↩
    _ tØó??«Ë%,øûj%ÇBfi£¤ü|ØhP??
?èQ$H
  ô?6?~Í?,L@?Ò4é$^3$§I?3ùÓ^E þ?Ç£?Õ?DÍ$^1$!B?HPKÈ,)1?K??¤??$^1$hgX©Ã???";>|ú4Ò£D?? ↩
      ä??}2d6Bc?\Ù?À¡Ëw2?è$^2$vd¾{&$Mi$~»?j?p?ô9$^1$Q¦Å??Ô?ÈÆG???(8?aâç?<ú$P%HA??È ↩
      ???@ÿÿK*HpùGL?0????§PdÅu??·îá?íôq§BpDÌIÉ,y$\div${FL?!pT?H$(?Dùéôâ$¡?¼"z*??©h ↩
      ?#HñÀÀäÌ?&DpéÒ$É?ûø\textdegree{}¶a¿}?ûÝ??8r6áÍn4?åê??Zapõe?L!$¨aÏ??3áÐ#:??
}0?Ó©?J?}??8t©s!?&4Ù2dÆ?ô?/ÝêAv$\div$j&\ensuremath{\lnot}?~/aÌY?töHÌÖ-Ü"f?Ç!? ↩
    HÁKFbC?HBd\ensuremath{\pm}ó¨?B ??~MD$^2$5
z*J5J?MCB*0FX?SDÉ?=?D~Möuº¶Ñ®Ë0];$^3$$^2$õóø£´$\mathrm{\mu}$$\yen$@)ôX»#  ÈÆ,ù?Ñ
2_t8ÐhO%??9ôôTjÕ¦R®b$\mathrm{\mu}$tjÔ©?~D¨???FM&\ðß=u®éÆ$^3$m?;/èiA?OOaIÓn?û¡éC?I©
À$?Â0Æ&10È?óih'Î¤? ¢TªM--%5*Ái*P~
BT;£Â$\yen$Ë*T7îÝk$^1$q§§§¤Õ«LýG?:ºùÎ$\yen$=õüi?îÊÈN$?çO?G?lÁ?Ån  ?24(ÒÑRIH$\ ↩
    yen$X$\mathrm{\mu}$5b$\div$¯ML\ensuremath{\pm}b?T  N?D üÜ?¤Ã}\ensuremath{\pm}\ ↩
    textdegree{}íÛM@??~Õ®¨?ÙÐf$\div$onh?yo¦Í"i?$^3$ÄcÉ?ÁÑó??4G?rT*P$\yen$Zªj]ÍlZªhé ↩
    ?¤Ñ?ìø?r0BpC
/n?Í{.?j·?$\mathrm{\mu}$££$\yen$$\yen$¢ÛQÜ(òõ¦??ü??#$aã?/<?ÓCC$^1$?Ð,Üa§¨? ↩
    ZzZaiÅ¤£?"$éO??4b?6~:xõþÆ£m jº`ÖÚèh6nßW]ä)YúF?
sg$\yen$Í(T)3&?
 áÍ?i©U$?H»ºÍæâoL-]Hª?.T)Ñ?.????e»§??G?5¦}6ùÙúZz®5[ê??SclêD"13$áÃ~F?èckwbÐ$\yen$N- ↩
     E=0¢\ensuremath{\lnot}UT¶©R3óâQÑ#-*,$^3$??½Õm®õóÆùùyyÍjgææèéê?´ô¶_íÃ?d?£å$Þ?
uÝ?J-mZÉñ?VÚXÓ$\yen$R¶¶¦®¡:Dª)?#,D?ñ»¨Ûn´3s´·Ôkáæ$^3$læÚî=  Ü?RðÚni©?ÉìVýM$\yen$$\ ↩
    yen$Z:?Ë®
2¡?§U»Ý¢ ?CÁ$\mathrm{\mu}$?û$\times$nö?cgklhäaãæi¶nÛ5MgðuÀYdHr]CÕÌCP¤ÌÑæWÕÒ¨F-$\ ↩
    mathrm{\mu}$UZ5ü[¦?««H?òÎ?ö??Nu[æcå7oË#=»|üÆymí@q¨áô}$^1$9Ö$!<\èVH"Û$\mathrm{\ ↩
    mu}$+V-âs-?$^2$$^1$mMZ$^3$$?Ï52?ùÐ[gé7ÑÌÏkèÀÊÑÖÓÊÇÊk®ú?à?§?å
á¡QTX«xt*?Ûs:?4ÆUïî7â\ensuremath{\lnot}@?J?dGxÙøÍ?Bq£¡Ï?1$^3$gàü¼lTaÊL4)©¨·vê?? ↩
    Á_ï¯Ý\ensuremath{\pm}\»wL?ÃHÝ???V?M©Îño½?yM[?ShÏCG>;üÍF]º
á©nK?"
ÚåÜt[/øÓJ?ÕuÒa)?$\yen$:>8$ø-?æ»SuÙxYaæ4Ëg£Ï+3WKC+õ?,¸ï%AH¨â?ús`Pú<}Ù¡=Ü
d*èJ(Ä
O=fË=îÛoEö?6y$^1$??òór$^3$u12\textdegree{}üB\ensuremath{\lnot}xãLpjj¨Ð?J·~K[®C
ô%+ù^Íªôàãd$^1$?¤Fy$
#ßWÃ©§Ó_???#ºÑo\ensuremath{\lnot}$\div$-~á7tiQÔ$\times$®«"¡CÆ¢
??Å?5
ÁÁ´âgL@gÍÔ?zw$\times$¸9rðòtÙâzr<À?´26?&ÝÕ&WGc¿D{ÃÝçV+UÎäÚ©~P?ª??î·k+??V?Û{ëÛììì¼ ↩
    ?<|O¯ý§©.ß
¼$^3$ÛºÕÓÚ-:~¿àÿ??dÉæ.¦õ?"Å?yË?F,Í?çõñäe´gÏÅô$^1$6`´?[C§¤¯ªÞ~?X»ÃPq-m®ò¦P?Vxì=?K, ↩
    F¯5òüÞFWþlýÒiöâ$\div$f)ci?¡?Ïb%bî?þ55Jë:4ÔÁôc3\eácééåãzp?cx|¸9¶?Ñ´m4Ñú5"QS[ ↩
    Ô£«_ÅãÛÒ SÄâ-VqJ~R},y¸rq?Auéôccf_âc~?$\yen$ie5ºABy?-G?¤í"î?&ÕR?xkªÄ2`s¤Á?v¸Wí? ↩
    ¼ÇeË?$\mathrm{\mu}$zpIM0à??®*ºÕ¢(árú???ÝzÚ?èLFþ?G$^1$Ï7Ýÿ&7«3õêEJ?»üùÉ·â¨?W?ÜZ ↩
    .?ö?Êã§Öò$\times$
NZ2r$\yen$¿}7g+Éáó6rÏÉÜ½eèú\textdegree{}Ç$\yen$êÖ/Ye¼¤£©ßWR®å{Ì7êgÅá\Y$dHñ¶$\yen$> ↩
    Ìeâñ3u¦Ê$\div$¿{ÚíÂ2Yöf¢ö?mj©ê©iª½Æ?.iÏPÖÛZW~$?ýx# {åõkkæ$^2$ò$^2$üº?"& ↩
    ëQ»Ô´íõªÉùþ?þ
ò@ÑëÕÑ"7ªÓC^IÉü7¾|Ç.4=XÝ{Ø?I:(ô
WqíÅ5OksÎ$^1$\textdegree{}?2 Î¢?E%¦?6ÈºÅðßz´pýMü¿íÁØÏÈM?5}?m?òêâÒÚ$\yen$?Mu?Aó( ↩
    ÝVó?ä_âæ·Ågõÿ?*¶t8!)?®ßpmT\ensuremath{\pm}ä/\$\mathrm{\mu}$?IÅ4<pò/06Ö| ↩
    E¶7ÿKO_Vóñ+ì?lÞZj:zJk>
êS5¾$\times$  u!ôTæbíË??ÐÕÖØMñ\textdegree{}$^2$u]Æóü$^1$V$\mathrm{\mu}$\textdegree ↩
    {}ô4ä§§YN-ê®ö.1?ÂÄÓ¤K#5£,§$^2$§Ç4g$\yen$¦ú^WÍo¿$\yen$$^2$%:¡V?d$b$\times$KlG·?© ↩
    ?¸ñ?É«$^3$4F$^2$3Ü¼3ÃÕ5îT<g¦8ZÊZêj$\yen$ö¨8ÖÖ?õÃR^l$^3$ùq[¿?;ëý]m(y]|õûÙ9XÍª? ↩
    Ê½EB]$^3$|^E
P?4BdÇ¾L?O?6fkºþ´¾jØø?hì\¶1?H?¯ßYÂhE?eÎ?$\times$øõlE¦©??çb;ìý6
m??;zýí*A}{¶?¸)ò?.Tê  Û~?ÓW ìÔBÍsç¿¿ÿUÿâÚ?£ÏéB¿{ðØ£) ??;oW¦ÍüÐª5¶Ë\ensuremath{\pm} ↩
    AÁª6Õ$\mathrm{\mu}$ïsð½¸cm?ÃÞþKðøìxIádl'`Ã+|½WÓ§ÞúüMim£{yÿW{???·?¿wO  ?Éó9¯ÿ ↩
    ??.=kY
```

```
«T7ü¾îüm\ensuremath{\lnot}çõo:}/\ensuremath{\pm}{??$^3$ÿ_Cª¦?X{2Ùz?]0cp6?£0írÒÌ»?\ ↩
    ensuremath{\lnot}ÍÕú¢Î$\yen$ßK|ÅÈS?5Ì~?î??½uË?`~®$\div$5~ùNéÚÃ+?AïVïõ\ ↩
    textdegree{}añS$\div$ëdJã1þÿ
&ÍÔW\ensuremath{\pm}^)|¯Ëºô[?=ÿ>;Ñy¶ã¿~ Û f ?*I??Ï`{??9f?á?iÎÞÕ?Âü¼rèif$^3$o¡&$^3 ↩
    $ÕyÕ¾O`¸Dy~?08¼åÜÅCõ ?KÊ¯$\yen$G\ensuremath{\pm}Þð:@<ü©q*W?úu?JÊûpÄk{$\mathrm{\ ↩
    mu}$?ÛæZ¯R[P\ensuremath{\pm}?.?Ü\{ß?pj?Óaü?Kbý}0ªÒ1?wùº]S@?ô£ëÖP½ï¡Ùê??¤?í?¨?$\ ↩
    mathrm{\mu}$ï[è« \lgÝÇ®,?ïO¡vLÉuÔõB?ÜÉ~æ®É\textdegree{}åRX?¾únÃ\ensuremath{\pm} ↩
    ¸?)7Ø/ù97??\textdegree{}z?2ÚS?w6»¤b?¤XÿoN:Þnð´¿F?9\ensuremath{\lnot}?/VDä0èléª
¿ý{sMAÚ8Ð©?î?s?På$^2$òíè\ensuremath{\pm}æÓ??õ?5?áVø$\div$0$\yen$u,è©¯'Ýë]$\ ↩
    yen$b´ÜïûeR®ábÁAðKÀ?¾¿z^?c·?ÇEPÐ#$\yen$\ensuremath{\lnot}ûuzE1§A\ensuremath{\pm} ↩
    yÚ ?é?Á?!1UoªPh^~9c<l?Ø
ÓÒÆEð|B!½¼ÈËbÅiØïI@<-?ê?û½Ó·?ÐT,ª?ÿo©vIk?Ô~«ÖuLXÿg?Îx2a,·j?þ_¾ò«?1E}òÝé§cïï ↩
    ?^}^#???6ÉbÎvý ö$\mathrm{\mu}$?ø¾ëÌ¸
\ensuremath{\lnot}WXï$$^2$ùnüü?m*{NËÕìXû¢È?dº
OfÁO«ì¼óâ8?w??Ißuº}Ù½?<;v]_?à?ÅyIÏ´e?vöÍ·kì2É¢\Å\ensuremath{\pm}ã«wäxÁ·Ïñà?Äÿ7?é$ ↩
    ^2$?N??åçïòýÿ§¡O¸Å §k©wàð\ensuremath{\pm}ù:?ï$\yen$âÞ\ensuremath{\pm}bÆæ=+Ý} ↩
    T¼nDo?^ïêðRò\ensuremath{\pm}åÅìtîü3AîãÞu.óE?Á¢+ïß)ø¾ýÔ$\div$w«Òé2¡`?#o£§Òì¨?$^1 ↩
    $?è»tâ?ÇC{Ýüp?®crgs?$^1$í·Åëôúwñóv^©ä6ÄïÏóá½ö:+øï:Ý&@ñ#ïõ®úÿüKûcð;?Û?öÀBþ  ↩
    MKÏüÀºÞÍÅîõ{@Á9Aý½?iZf?R?î?yº?_´Å??Ï$\div$ð\textdegree{}ß«zçëü¯ÐsÀx:¿]jv54M?*~ ↩
    È?X*)ÜûôÝ$\times$KäiâE·è\ensuremath{\pm}?c¯Øý¯81þ?$^3$2éV` ;?;Á?ú/DYú$^2$0[ø? ↩
    ½¯â1®ôôðþÉ\&??É©P$\yen$?Dû?uer¿??ò?&4$\div$???oO?è$\yen$?§áB?b?CpXÜ-:$\div$»zÎ
¢ãg9QqÉ\_$\mathrm{\mu}$ÔÊûNP?c@ê(ãÊNÇd+/ß!º??!7lí_?ììõ?ýðEÅH$^1$Ú¨¤ñ??yp0\F? ↩
    ÁÆ¦»Ëivz|uÿ½-va«E*G9@Çáá(;{@ô'._m?ÈãYüºWª#ó´Ô'b¦'S\textdegree{}æ)EßÁô?F.».êxüÏ` ↩
    $\times$íÕ¾û,ÏÝñõÐØKW·ÇN$^3$GÂË¼òl?îäÊÞ0$\div$Vî¯:ïôõDÎJÞld¶?¡íz\ensuremath{\pm ↩
    }ø?ï)$\mathrm{\mu}$2î¶~û=àà·¼ö?$ï:¿Ìõ(ë«F?7u?Þyôº???j$\div$0|9?moÜK8?[m?¾$\ ↩
    yen$çWOtCÆbiP9Çç¿ßcÆáFyã¿Æ¿ïçEEÇÜç¯ïêuîòßJ\ensuremath{\pm}å1öîk$\yen$è}} ↩
    êïu7x»üo;.ßÝån{Û?LÔözK¿?3q9LxÌ~?Åt>~¸v<ßu7®?5$\div$$\times$wÒóËÜõ$\div$½^ ↩
    ½ÝïcùÀcïMi¶þ¾c?6~ÝRíí?M?¿§Òéß'???[»Öìõîü?9üÀ}yû]|håE]îÜT?óõîú]MGK?Çec§ÒÏSI$^1 ↩
    $ööÚ$^1$|ã2?[eËÅÂÿ}
.ï{cævP~ß½~é`¶= §PKn$\times$Ë??URú\textdegree{}Y>¡MB?æ?Þ"VÙ8?]ðó96Ðgéª Q?83??u£Íºu ↩
    '?ó¯?6y7å?å\ensuremath{\pm}èr¸?
'Td?Ò?T_>ÿ?
HªUü?k2&ÏÔÈcçø¿?7¡À?®Ë~dñôè]èá´Ä§ÖN:ó+Rülx?Ï^ÃÏ$^3$?E?2/$\mathrm{\mu}$
?FqË[üDyñ(B¼Óo/?
ø©S<?¯ílmZÅ#æ8h?ÛÖ!~ß+50ÌÚûv?3Ï?¢?xü??$?û$\mathrm{\mu}$e?nÐá3Ê·O5r$^3$H&\7ÿïY¿ª. ↩
    G«Ò? $\times$ñø~õZªêBäus9Fqû$^1$ø2g?Ý:ì:Ñg?&" ¡ònxl,D[\¨Ñ:h>¤ã$\times$úØ$\yen$L
Ïút?3Yáá"$^1$þö©¸?4q¨!??gmìgäÉ?Óúu$\yen$ý¦6:8¦®â0àQplÅ<_a(ôTg m:Ïg2?r
¾Ç?8Ùåc$\yen$?#ÅçÛ*¸ß
|ë j  ¨¨$\mathrm{\mu}$\textdegree{}Ó*,8ÙüÚÙlÞ4gC´zî;?ª®V}?»Ô$^2$?NiæÛ|¼¨pd?ióºÒÁ? ↩
    SÆÝ{|;]¢k®?1ä.õRÊN_eî®K=½$^1$D4¾wùø?ó2ÚQG=íÜðù+Rq,UÿoX¶¾Æ3=¨ÃÊë}¯[_ÅÌg$Óâþ?·_? ↩
    ÁZâ?ÚègPu«ê4oþí7ÒZd5D1®w$^1$qÍ3¾\ensuremath{\lnot}>q?ýj?Rq«é?1ÁÏÅü$\mathrm{\mu ↩
    }$!x&3Êh?}$\times$½ïñWÖ?ª«?9ùÔ?çÕ?ÔË¿¾¿Û4IÿÍ\ensuremath{\lnot}Ö?>;0Îð\ ↩
    ensuremath{\pm}ö½½ä
ZaÅ?bD$\mathrm{\mu}$?$^2$ð<?®?d4Ö©|Þh{\?ö?$?$^3$ûJÛ?I4Ô?jiÒ7§$^1$øÞa?S¾ºÞav?ý?ÏÎÝ? ↩
    K?7V¡ß¿5,êÛC«w?7í|Ý$^1$ ?´ÂÜ_Ë$\times$eR9ëÞ?Ù¼Å<ØèüÖ&ÙwåëÝÞyK~îÂ?:»QrØ¯$\ ↩
    div$¸ágú?Þ"î@%4=_ÓËÞü¼NfÃu¼w$\div$tþn¿Ù½ªø½Û
?Ög\ensuremath{\lnot}ÜÎä$\times$Z·ûýÌ$^1$\ensuremath{\pm}Ï'Jhñw?N·Õ¡¡tjqìÐP?$\ ↩
    yen$?6<ê»#í;lõW.?%ô¢UÒCü:»À&=Q3g,«ÄB]ÅúÞ?y|\Ý$ÀÙÊdV~K?ÅÕÑýGÕOû'Î?Á¤#»cm¾¨%ßu8U ↩
    &#gxu
?\ensuremath{\pm}¯Hßèðùt)?Ù" A
$\yen$IP;ý®+\ensuremath{\lnot}?ëø[¢ÝÞÀ&jÃ(ßS7$^1$@%RÒÃÇÐ?D~;?5ç6Ùc?Qd?Ù$\times$¿Ã? ↩
    íÒzzÝ¾ÛÚvs?$<¡ú~HìûðSØÓ?ê$\div$ïÂ,Ê?7ÙÓ½ïRÓ{»òP Å1j:??"$\yen$âÏ¢$^2$·+?Çþ¿?¨í}þ ↩
    ?¯ã«Ù¼?¿·b?^Îz*e$\times$?}rp42\textdegree{}RÜ/Tû?¯w§àæT$\mathrm{\mu}$ù®ÿU¾Ë??ì ↩
    |<éKk9?jèºÍ?$\mathrm{\mu}$)îsmd}Wwl9Ì}S Ú?ÐbEkÔÖ»Üöº+½¡Øm?\Æàðéyl$^1$}Ú??Vï»®
h8x_¨Ë?y?ÁK?.¸i½~¶Éöõ¾\üÿLÜ?GêX?åO}?ÝåË?R87\ensuremath{\lnot}ÃZ·Â¿Ñõö1]Ï_T? ↩
    ½Õ»úûì½Ç«ÒËRÅÉnj.UðV+Öð,åÞõop$Ç·©+?ðéÝêT1ù.Pý½?\ensuremath{\lnot}`Pò¾?Æ?ò+?M$\ ↩
    yen$Ûüq¡?MÁRzSÏ©w.Õ?~óè»ó§`ÇhEã·aZ{¼???Ë½ó¸2?ý8aD{òô§êú5Õº¿5æfñ?ËaÃ$ý·'x?Z$^2$È ↩
    ?¡ÂN?t_5Ùz‘¢êý?®ªÞ[
Éf?K¾º??Bi2Òp?g?:ÿæ»
g?Z¶ýþÍw/ä[G?!@Ô.áRI,?vz¡?<ìÖÔàú¾k·«7Õ)àö{îEã{ÖT3c?lK^?ÉE?lél?/-\textdegree{}$\ ↩
    div$òËkØéì?]IG?ôî´Usýì$ê?¶$^3$ÄÍB$\times$t8¼M)?%<ßóê64$^1$ZÿÇïEÃa¾§!?!+$\div$$\ ↩
    mathrm{\mu}$EBC)Û?¿$^1$"V/çÔÈ/jd¿?åé??  ÛU?$$\div$ËMo·*@Çcyº Þ95"$\yen$.Ëòò? ↩
    ·aqPW^?$\times$
£Ð?!?x!ú1á@WOë»ÍK^~Ö·¸úîX!?â$\yen$«©*TB$^2$£1?RDY¡??»Õô?X??î2K¯r¶~^\ensuremath{\pm ↩
    } ?P¶¨äÈ
@íW3a<=¯ßÖjIa¢¼Þ?â.¶TfDtÞÅÊ?Ñ
!RJ8ÒFôÿ$\times$máÓÅ%7îßí?
F-MeBe??T¦?8Æ/$ÔRâc^ý#$#?kÅènV$\mathrm{\mu}$?ñ££X¯yD*1+¨Ã?%?
?Ã?Ññð{ûc?éß/XÀ¢?4yº%??üòahwEKQ(??að
!1?$\yen$þ4Aå¡½ÔÄk?4Ç2q´¢$?=Ð?‘Â!?"$L#I??rÑ»ÉGdÃÖmªôpd@Û?@ÉÃ??|Ár?A2JiÓÂ#+"?´ÈÑÈÁ$ ↩
    \mathrm{\mu}$?TYdJ:t¸ÓÄ(‘Ábæ6??Î@débE$Ê??ó?E#n\$\yen$K0é$^2$fÎ?"åÊ?*‘Á?‘ÁC& ↩
    ÃDx´ôH1¦?TÁ20¦Ç#,|$^1$Ä???ðò@6\ensuremath{\pm}QÅ??¢æI??t8¤\textdegree{}?$<y?IÈ ↩
    ?4á JXÃ?H?((<\ensuremath{\pm}bæK? P?P?$??Í.?Âd£<VpÒÂ*DáóÒÇ?0\Rd?D?òã??ÝÇÏ? ↩
    $T©RCåJ?©b¡?4à?îP ??2J??y?Ê???Ê?âcÂ??sÃ?Dxà?ú?ô§üÊn?\ensuremath{\lnot}Pr%ÉM??  ↩
    \textdegree{}?=4??0?n8y oË? Wsñ?$X¡Ä?ã?????5â?Í?D?&‘2?ÑÊ2P\ensuremath{\pm}1å?? ↩
    ¼Ò?Å’t\ensuremath{\lnot}Ð
@ÉàkÏ?8???K,T2¤J.@~DN ?2?Â?8aÃÈ?$1?6?·?ò$î<$^2$eÀ8‘?rÄ?#,?4bÆ?Ï?8?’L
¸Û?2Dð6‘f ?,Á0£‘\textdegree{}DJ??(SeÈ&D¨ù@b£Ç??v@??zÄ21ChR¤O? P©i¤Ç???ò~1|?L¸?áb$\ ↩
    yen$Ë(@?ÐÅ¦?i?c1a½?§òA¿Ë8?!B&
·??1PÁ£FK???/4\ensuremath{\lnot}h.vZa$4X&N?^$\yen$È8???$^3$¦?)$\yen$H
|ÉÁ<|íèâ©2~3?TÂäMÏ*<(óH<  Âä?#?FÌP!?¨Úí´6$Î6HqI\ensuremath{\pm}?L8ÁA,H¨ÆC????0= ↩
    ra¤J¤C?´=XY?@
<(ñ6e:=?Âs?uM?©R"?qY%Ã?N¦5?þ¾O;o?Dî.Ô-¨ÆÐQné(?">R\ensuremath{\pm}TQQ?©XA?F~î$\ ↩
    times$$\yen$bòÿ7M$^1$}ûsgE?4»Î?õZ?STTRÑ\ensuremath{\lnot}¡¢.F??Ç$ÂÎôv:¦6£R~%Np  ↩
    ÂY$\yen$?y<½è¢A^¶\ensuremath{\pm}2?!û+/J?ã?|‘$^3$??\ensuremath{\lnot}?%aÁ:fZ? ↩
    âYòØû¾º?J$\div$tgú,   u\ensuremath{\lnot}x1½_gOþ½Z
©??(¨-lV?IWÍø$\times$Û¨~þx¸e_çéë¸?zNF?ô?ïeYjÙñT¦à
y
jõë¸5t6aC)¾Sfº?ç?9Ìº¿£=)VRSþðøàÓï.}ªb?Úo_HÚzÛkm¼?!ÑU’/¯Õe§.Ô1?£AS>îy?-¿
mQ}Gð‘Eo$^2$4-sæhV©
¿c§ûi½Ir)ê!?(ÝÇEjåÕH‘af9}
>«i?](?ÊNwÏwÛÅÕ+l,ý(Õ©á-%¦ÿfªN6?G®?fë??ãïîÇñuz®´z»?ôÊÍõË½?ð¶|+Z£x8:áë·: ↩
    DZËeöoõîþÏ3C+½z?JªQ& õ?î?ýXìOO§Q»VÇIN®Ûr#pP¿øæl$\times$ïhÓÒÑ£
Ôq**nn«Qj$^2$¾g¨á¶ÌÁÃãVârWHÒùîúW¾¨«kU\ensuremath{\lnot}©A]JEññk-ø[áKú¿?6$^1$ºSÅÊ[ ↩
    Éæ0?lïï$\yen$ü$^3$/¿?ð$\mathrm{\mu}$SúÓ\¡©ÝðywTcc$\div$¼?M35à1
$\times$È½\OË©Òé_e$\yen$\UP«,??~_e~ýL?æóÈÏ??\ensuremath{\lnot} ?:$^3$ð1$\div$D~
&ú^<$«EQYL0Ð¨ÖÙ]r·þ´¿7åÝ?eâJ?$\div$E??~îK$\yen$ÛeM¿¡?revB»OJ$\yen$s?ªf2ÿ~LÌ?öðG?9 ↩
    ¸??O{[¾ñâ5¿¤?ÊÆ~´-e´vÿâÜôì?ªñ??£@%ÏÆÇä§?ãë]ô¯;p,?Cg¾?ÖÌ7‘*Mo¿Ä¢ cËÇî4´I·/´áP?: ↩
    _/ØÊ~ÀJÊÅö+à$^3$[UÆaKº?$^3$<~Yr¢?ÝbÇû??õ?ºÝÐÚ?;{:ÅÛÙ[bÔ½ø?ÐCôõÿ,ï.?2*Ü¾?Fßë?\ ↩
    ensuremath{\pm}Öó?¦?ßCeÃ??êÂ¢Êåw\ensuremath{\lnot}#ÎÏSÁßôélN¦«ä?/Ø$^1$«tº+U? ↩
    ºèUËHdkXW)óõñ|3$FV,ÜKyb·?[ý/«¯2É@¾Òäë®  Ià-H»¨ÈÞoó’øðfÎ¢p3Ë¤\ensuremath{\lnot}À ↩
    ?_K$\yen$û ©Q¾_YÁa]´rÍZÅË¨?·¼êø0;7$^1$Û!SomAøx?à|Û¾Ø[º~??¶Ï´;*ÊÎ?¤ù¦$\times$^ ↩
    ÇõòdEAUee$^1$fÍ¸xß>æ/î/RnÇßZ·¡Å{\ensuremath{\pm}RûE!vº¯?c$^1$5]Ï?q?¸1?º?Ò«Ë] ↩
    GÅâü??½BþJ?û_§S«Ýû/[I#Uqd?7Xò©ô¯7??KÔ$\yen$J¿i}G¸ÂÛJ.?
 Ó½?-ß{·ùã·w?ÛÜ?kïé\ë?©£[Ë´\textdegree{}ç[V¾Â$^2$éhÛy_Mß$\times$Üì$^2 ↩
     $ØE¿_¸ÌÿÊãLþÐæ3?25}?õL?Í!ÕÛÚ;~?ço"$\mathrm{\mu}$ïäzÕÌ¼ÎO$^1$¸#^¯ìþ?tõ\ ↩
     ensuremath{\lnot}?WûüºZõ?®©ë?è»éÞvr‘Ì?w#qñ\ensuremath{\pm}å&Ï½ënk?yj?q>>?ö<Ä$\ ↩
     mathrm{\mu}$$\times$+?\textdegree{}4$\times$Vÿ¯wöÞÞæ»\ensuremath{\pm}CÝÜAxt?= ↩
     ßë?Ö2\ensuremath{\pm}E»???@[?i:Êàó$^2$Àý=?´ÃU\¦?g0öú{}]ÌgJ?íZö<Ð.bÃYë+?? ↩
     ã$ü_8B¯û|?1??û?Ó¯w$^1$Á§©^ÇüêK1WU¡TAÓôAümÇ$è\ensuremath{\pm}ÿ Æ|W*?éô£o@- ↩
     §ÁÐbÆ®§z¤Èy]¾íïW$\yen$Óî3?ÝÆo$^1$
ËBôýOõûÐ®$^1$bÇßÿ?~
ÊóÓeÞý\textdegree{}?Çúüzf=e?¯ùºXïù?aÆÝþ$^3$üO{úñÿêzP?{À=¿»?Þê·ýz>?ú-?Ñýc{?¨åì?æ¶1\ ↩
    Îß[éüzÝ+ÌxÇ[¸??ÄÇâà?óÉôI¯çûºãýû?n~é¦Ù^ûë§ÚÅ?¯_ÇÜÆô~?e<&_wôño<$^1$?_þÆñÔ+¯Ýú.û? ↩
    ¯ÙÚe?%jàzË??s0»?Ô<þ?Ø(?\ensuremath{\lnot}oE¾)P1½?NÇñÿ~MAÔ?ky,Xül=åõ&s¯~ÌoWÅçÿ5¶ ↩
    +NÙïõüýËßÂõ$^3$T?ÎàÚ|®]ªW$\div$ý®â?{?õ®v,)ÌwêeyÕü|¿·§X»?’??/uNWCÇÿ??õ’üBù?ãÌÕó^ ↩
    þ?Oñ}{àg 4ö?ÀbßÙ#A?ïõ1?Àº?¿Ã?6E"EgüÞÞv<^n$\div$Dc~0ç~Çüó¸5$^1$ôþý~¦â?¡£+$^1$‘ ↩
    Çßª?kk?ÃÜ¼û|1¼7úåQUï®X1ÿ?$\times$p)BlË·ýK/OPCs?Ïø¿$\mathrm{\mu}$$gð=?$\div$Ü¼ÿ ↩
    ~[ÿç??t?¼???u¿ÞÑÌÒeûüÝ}ÁóÛé?þGGûSm¿}}Üð^\textdegree{}?F?é?ºîÉÍºàW$\yen$?SÑü^ ↩
    uÃÃ·}¶¼ßor’}ÎwF3§ì».Ýç£&ýpÐÖ[Ûr¾.eÕ$\mathrm{\mu}$‘õ\textdegree{}ü]?~Þ9ÙÙ·?Dcý® ↩
    ~+IçPý_¯$\div$\ensuremath{\pm}áÍÅÈ?ÑWoø¿??¾\ensuremath{\lnot}LSl??îìî
?U7Åï?\ÃyÖÖíáó}Ý¶ÚæO(àÛo$^1$Þç¼L?½?M$\div$ïyÚú®û¯^À\Ä?s»ä$^3$åÕÿçèî}eçM??¶ã~¾Ùh$^3 ↩
    $ä?ä$^2$¾üÿ.ßËÒg?Tû?1ü?oë7pÜ?ö{965Ì¢J·z·Í¸$^2$«¡?f¶#//åü~}ob+??<?5ï½Ë£-g?ÿ?fõ$ ↩
    ^3$|øGDSicd?J«ÅFZ>¦??·Þþ:Ý§Ñ+ø9ö>ý$¼¸-MV}¿çÉyâÖÖo%?í? ãqm*Ö?##O?
þ$\div$Á{$\times$¼?êv?Ð={NUÉFº?ï{þ>$\times$ÆË’\ensuremath{\lnot}[\¶çÚÞÕQ)?¶&??¿åû ↩
    {?$^3$?_Ï??$^3$âñÎ7g{{äý¼:È8¡]zÚûn??¢ØR5\ensuremath{\pm}\textdegree{}1?xð?o£& ↩
    tY»î4?û¾$\div$?:ÈËò^$^2$î~$^1$Ò¡G2%%uBÕÖõ
?!6?6XÙ,¯ý?gA?t\ßsÇYí-êî_yïp‘ËI*Álj·v[úÚ??%EoÑ??-æq(ÒëùØ|?$\mathrm{\mu}$$\times$*] ↩
    b$^3$eàôxü:d?]?j+©*,Ò!24Z8ùºlò3/ouÁ0$^1$zö?ðotñ3<$\times$Ø^OFÉA=D$^2$ªP$\ ↩
    mathrm{\mu}$õÐ??vÞ6l¼($‘_Æäs?¾´+çeçõm>}U%Pªý{
?ó?-¾£M6¯;Ò/Î.º_ËâToìCkÊÿ?Í?~CÏ©$\yen$¨$\yen$MM[»?2cN¡áÎsKß+Ù¯®á]rîìÎ¶ÉÏô$^2$ÃóbÀ ↩
    ’<A?{\ensuremath{\pm}DUYO@l~Ãáéæ¸ØÕÍÍ½ñ?ò{Æ$^1$^¾¦ÆØþ¶N?,&Xp
??Ú$\yen$«?$\yen$ª¨?é$¢j4ÕÙoß?õqén8w^íO$^3$[#?7!?¡2ã??QELZÚ3åMÞÆn$\div$Û$\div$¯Îz: ↩
    ÓÙ¸äqw‘8_[?+??,???q‘$\mathrm{\mu}$J-E?¢cÇFo¯
Þ$^3$ÝïFÔÇä$\div$ë®9?ª--Î5ÆÍ¿ÀÆôcK>Tr?jjtçD§?|\ensuremath{\pm}A©¢ò;Ým[Ö[R¶KÚðm$\ ↩
    times$®Ü£?3+CÓË?cä&¤?~@ÑÕÓ??ïc7m?kÝm]?òâ3É~Å½·"ê¯r&Nû/W§
Iò?\textdegree{}Z?ÐTt?8?«¤úT?-Þa$\div$qà¾MoÁ¸åÚWñjabgãùp\textdegree{}\textdegree ↩
    {}\/,Í--*ÃÁ\ensuremath{\lnot}©D?fMw¤î,W?Û?ü=æò_´\¸?;~¸ûOSO3,l??? ´‘?e?¯Ft??ßV?h ↩
    .=$\times$ÿ¿?Ñ~?¿\»Ýº$\yen$¶´-üàä‘f?A8U¢1D?Øe?DÖÖ?îËíó$^2$Po$\ ↩
    times$pWÜÒÚÚÅÆiæôãâ4&tq%?êHV~Cg
?¶$^3$ý·Ï?lB¾ñk:Ñvþé}%mºLü]?^y?Ç "<AU
 -$\div$¡8xùr18}?ë$\div$ÑZø??ýU½$\div$Û«Tº¯o?S’ÎÏ/Õ$^2$xÄÓÔÊÖ,?J43óá??«Ï¿ØÙ|èÔ9? ↩
     lÖðxªêPjä·ô‘g$^3$Êx!RÑBTÒ??2N;¦î=?ÔI>LC??ØW$\times$ð$\times$($^2$·ÒÐÓe0s??e@ÎÑ ↩
     ?òcákj:|ö&Ü?$^3$}?ðJìJª¯]ÃSkì®$\times$SÏ4ÌÒ"
$^2$\ensuremath{\pm}1Òz?¨M?kSQÓ$\div$ò£ÀY¯$^1$?.VUToyBÔqª¦eëz?ecaë?A?´Õ0 gÃ#6Cî¡A ↩
    :#éÁ2ô½!©&mÅÒòäû
t|yXÙ?M?.~ZZÕ?@tÎè¨ÃFs\ensuremath{\pm}
<R?F>_?bVTË?ñý5ÞÊwMy}LýYSÅÉR
¤Bä?.l?WËy?4¨ä?þl?óyZqé)Ö¯^/?äWºY~Köx¸?0~QkT??3F??(‘?Qãí*?Ø$^3$r¼Y@ë?%Uî}m
lì¼öWØÙ8®U(¢ÔR?1C%?:hÉÏáÄ?\8[Qå2ójNÙ-e´¾¿H$\div$73É¡Î9Ó??SQT¨#8©óç?j???¶¿?<WÍÈ¿.ó{ ↩
    stªÃ\ensuremath{\pm}îfg?K+"AÒÅÄX¦0á?4ù¡Ã{??q?Ô ?Eèj>,??$\mathrm{\mu}$VWBÒp?$\ ↩
    times$O??I®K£¡B¦ZBòZ~|ÙÃäßAÚ?ñÜ?s?Èô¼’®õ8$\mathrm{\mu}$6Ë½k?.óoQ¶?f£<?É?Dx Ó?‘ ↩
    Gåë½¯?Iæ2pA$^3$?SÛÜ§[Ê\ensuremath{\pm}ªÛÑ$\yen$?2e?&¤\ensuremath{\pm}P?ógNPnÏ$\ ↩
    mathrm{\mu}$¨ä?gÏe.Dô???Va)??Õ¢ò$\times$¤qË?+Ñ¤?¢’QÖï?È*l$^1$ÃeM??åÄ?PÌú!yò$^3 ↩
    $CàÃØÕÏ*$\yen$:ÞE]Bÿhî®¶>?\ÛýSGL$^2$º~Á.‘g§;"D?û0Ê’42$\div$þ6ð3Þ£H$^3$¾$\yen$] ↩
    Ë©$^2$ã’E¶6‘ÆF$H$\mathrm{\mu}$’6Æ2@‘;[©2F??Ë?.?s$^3$YxÚºÏpi?¶Öõ?^õ] ½?<<x$@"Êò ↩
    P?$\div$?iðÃ¤By\textdegree{}êaPË*Oc¿ßW*2??ñílØr¨ãÃÓÉÎÃòæM?ÒeB??RI?,?
Ö$^3$Ù‘
è??8îàëê$^3$}=§/{½ø9*£ÂÊÂgç½Ë$d
$^2$«4<2G?"x@É?#&?Ú»r?(L¸X??(?è*x¼ZÕì7¤à7¿ÆÅð$^2$Hié*ý? ^lp?È?ê$\div$Q«$^1$sX93y. ↩
    NPWr?XrØWO?ÓÁÇÁðbÉ*LòkË?#|/?y¤ÝºQ$$d#$$dd (??"?
 T~ ?®®W]^ÝM¯uÝÚ»$\times$wzííNÝuîmÖëÎ·6ÕªÖ¶ºJE(
(~
ª7ÿ~ú«=cÕÀ¾´$^1$Ï8i?$^1$mV¦!å¢H?o?ÞÈã?¶Íì??$^2$Z}:&VØ$^1$Ì;0PÍí|ïÑ¿ÚíÝ}L4ÇÏ$\ ↩
    yen$ð¨<J?\textdegree{}?©??IÏ>f?{?!Ä?QP¤G§5Öè2s}a ik]ãþÍêrâââ[Z¶
j"WAT¶,D?Ï?l?7ÑkÜdõã$^2$¡a?9Ø8Q$\yen$uU»}wÞý[õÅóØ$\div$0m\ensuremath{\lnot}ØCÒ(úº ↩
    =?òÈ??Æ¢a?äiÏ?ÎÇdéÇyÙ©$’Í$\div$ïqZ\textdegree{}Ár]>î«gù¼êº-;7?7QìhfÎÖT¨}?WÎá#R ↩
    }?´úg$^1$Xíä$^3$N?ç?ls0ð_
|Zzÿôó(?ÅÍasÎÄZ?¦ªJ1ã;îâ)ÔIj?’0í&Ýì\ensuremath{\pm}Q#@lÛ®Î??<?R2´??mùÿÒ¡  §,q¯9·ït ↩
    ?é$$~C ?m?&?ÌÊÉJ®ÔH<fËl£?Z’ÄG9~Ã+?\ensuremath{\lnot}d+«üJ¯#ÎÛ«C?3’?ÒdùÖyåt$\ ↩
    mathrm{\mu}$2ç?’7?æ|j/O3á!Ñn$\times$3¼Ã?$^1$£æNHÊCÄÀûr1?FEA$\div$ªë¿ÏÐ¯R?»Óùï;Ï ↩
    {å"¨¦ÔÉj?þ?%zsÉ¼%g‘gff2î0x?$\mathrm{\mu}$:~V??aOm¤44?zªÏÇèXKÉ@köõ~{®î{ ↩
    ¡úíréåtÅhA@1ô?º?¶ÎiÁÉ}1$^3$FP?î]0ÌÆa
?âí*6ÞwàC$\yen$Ë?ê$\mathrm{\mu}$û‘;ve$öÃT?\ensuremath{\pm}VÉÔLf?Çmû½ÆoÂË?e\ ↩
    textdegree{}zýÖ\ensuremath{\lnot}ã#¯ØRíüÛ?ºxÎóqz?LÞ¸x=J ?$\mathrm{\mu}$hMFÒ?Y9% ↩
    ¨Ô<sËíhø¨azæ‘5q??½UE?ÊÇÇ?ÁrË©sö»pëHòº9¤
WM?¨h?H©,X?ß8ÊdçD¨E?îß\textdegree{}ï$^2$a?GQ®£$\times$ùv5+RHAuÛèössY$^1$ÒE=@¢WÂ\L ↩
    [Dæäi%JQi´~gºdÁ«¼üøP#HiÙÛ&0
ê5Õ4?_Q2Æn½·gºÓ½a=??£P?Ü
  ´GL"U¨Bn$?þû?ó·ÂÉ?¤Ký½¶Îr?$®¦¢køÿ?ºgO?û.¾?uh?mM?ö¡\Âs´J:Î"$\yen$?sºP^¶ÈÇiåÌ?f? ↩
      ébÁ»\ÓIg*©ê|3o@®F?|[?îó\textdegree{};}nªUZ~AH??Dpòuz2??\$\div$»Îóò¡C2xË? ↩
      FÝöLt$|*¯?a$^3$ÿ=ýTÒ($^1$ízß^?F$^1$ÙæRÐOê5?T3¤P½!Á??~ùÃ&?:|ü¢æ¤#=ÈÉoÞg??ÝnÒ$ ↩
      ^2$«y¿Ùê?vÛ?$\yen$ÓÊÎÎ jjUR´??OáÉ#:}:$\mathrm{\mu}$2¡>vÕ?ë/_h5 %?¸VbJ[]$^3$ñ* ↩
      ¿?þ¾uAy½»{¯·¼$\times$>BJbi>$\yen$X¯ü?rH¤D?j??íÃ,\VYÎßÁ‘äVÿnKf\ensuremath{\ ↩
      lnot}K;Q´Úí??àª[%£ÙèÞ\ensuremath{\pm}pè£5T?S%Lv6ð·?*åÕ!ßÎÊÃÅÊÍvö?=1Ã?{$^1$9® ↩
      ?:Ó
.VÖ}ï’o=*eÞf7Rë$^1$øÂ?z ÙTÏÒ?æJE?)Bvë5?mÞ?PiÐ»ó$^3$ZÃ@»]UUU¼ÞÕM?hqzx,ûîôN¡¤???
øÜ$^2$)5
S?ñëf?k.þ"¢??6?nÏ1ñõsÕUûm¾Þ$^3$ÂC$\yen$ØÈu?úÔ?©ÀÊ=2?$\yen$FJ?$\mathrm{\mu}$dÉ?¯J? ↩
    »9 W{#’ºÑÄ\ensuremath{\lnot}Õ=UU]lâ®{Ìñ\textdegree{}?¾+H1ïªFlðÐ}?#©õ  5??S)À$\ ↩
    yen$þ?FkÈ0âdä"?B0dÉì?Óe_aR\ensuremath{\lnot}ô?î>?$\div$]â£Wª¢Ô BxJqcÉ?"LJ? ↩
    ÏwYd}$^1$?ÍÉ#J???{’?\ensuremath{\pm}¡!+?U¶ÙÔÒëfÐÄtÙ¶oy\ensuremath{\lnot}? ↩
    GéÒ$BFùÍ?@\ÌÂ$\yen$?%Þ$\div$$^2$$^2$3hò)r!£?’ÙÍY
t#?«i[MA6¨Óæ$^1$L$\div$$^3$aÉiÒ’Ô?g¤

»@lÜÒ$\mathrm{\mu}$ ¤‘ædcb0ÉjüÉôi¦C!?p<·$^1$H?$\yen$ª«$\times$O.VÔ’=}¡?ò?ç¤´Ò? ↩
    @N´éÕ+Ô?Dé¡»ÍeÝkÞo?1?$0%$\mathrm{\mu}$ËC,çñ4Èu3Ôt´?$^3$$^3$JEÐzé¾s¢?7%§N¡*$I?¢? ↩
    VBJI?ózOÞ9ußmï?B?t?48:?4?»ïwhEÔMÑÓR??øHä
~ñ¾k??C*§P?óÆÏ H¡:d¨Ïh$\div$Ú$^2$Ê?BÛ>!?iP~û¾ÃÖÁûs´Pl¶û{
êE??$^3$Z}\ensuremath{\lnot}?>0y?¦gT$^1$4c?¢C0DºÙ jXÎ1ÆÁëöX7=5jÒ.Süæ87ø,3b¦ÕRÔxU? ↩
    ôçKhÁ¤_M#$}?ØÃ??>6$\yen$R$\yen$j%$3ó{\ensuremath{\lnot}r;ì$^3$??ÐsÎ’ï;î3>Ü?[A?] ↩
    $\times$T$\div$$\times$O+HxÁOhi?24a*$?R¤?ÑG0ËæùÙ;\ensuremath{\lnot}»Î?hÅ?$iÄ*? ↩
    ÆùÙóY¼]%VÊ$^2$·eUKá#?itáÔ ?TjÔJÊ§HéÃ’
Èi@wÔ[ÿ39Óý#?Ðô¡>tç¾åÃþvvwuÌ?§$\mathrm{\mu}$??l¶U»

J?Z9îÝçÃ=(ZÙ}N¡.HÙÑt´‘À?ë$??kÞ@( ??¡@ÏxõëS\times$ÙïóßgY~éÈ?ò5szêÚÊªª!{? ?SQJ«VTª ↩
    ??â?B?=-(Ñ¢AñËÜÌÜîû\ensuremath{\pm}?$\yen$???Z?ÀQOàAÐ7!G9AD7þmrtr:‘é(x?ò?iR#B ↩
    ??1?&1ú!P‘èg¼ï$\div$ûî»xL’?H$^1$}(z%CÒ?W=þ?~ÔøZn¿_MK@»PL?Ã7¦<Fztºt2&d?d?;8\ ↩
    ensuremath{\lnot}8?]ÕÙÃ¼ø!ÿ?Ât!B.?íóòW(ÙàÿTÉÔ¨Ô«Ô?p?mº?ÆÎ@7?ñù?&?£?&Xpá-Ðÿ=ë\ ↩
    ensuremath{\pm}Æ]=55ö|-? ?r$‘ú?cç??é*¦]^£P©^$\yen$2?’ãÿ?´éRJ??n?2X´m(?ÀÚ:??;?$ ↩
    ^3$B??0´ag¿?@åü-?b{cBP\ensuremath{\lnot}$S?Ï¢G+*0-:D¨P#I+©Z$^1$r$\mathrm{\mu}$) ↩
    D2ÈÉ??Áçqs½??$^2$?Z0¡=|ìb¼ý?ïß¼{??áC?o$\yen$?H>Ú4?úóèQ¤L*t©?¨R©BdÒ¢0ä?7¦’¾8¼$^3 ↩
    $AùI?ÐÊÑ~£~)¨OàgÀ?=ûçðFD£?H>¯$\yen$?!i  ??¢."G$téñG?¢x?F??)?c1?§n?ütÆ?Ý??§??¤?O ↩
    ?$^1$?ñaèg½?ÐÂ:?Ã?R??="à8ã>|öLsL??B*?:?l:làâH? ®H???*‘nA½4pQbñ¢Ä?ó?\textdegree ↩
    {}Ý}?ï>L8¼h ?$^1$qb/ h>xf@ÙÀâ?ã¨?v??jxWM?NH)P?Æä ÆÇÍ2hjñi¤??/?¢¡h
Ò?Û?.æB$¿B?N8ùÚQ
1 jDéÓ#C
?ÉÁÈ?ñÑ#D4$\yen$?ÈèP4d?MBùÃ??ýÚJhìÙ"(¡A?»J@ÑÑF?¨eATBò?%??M8paÁÍiÓ?>~HüñÑÑÃ¤ãçÅrÏ)3 ↩
    £Ñ>ù??Ó3¦4L¨:?¨£Pô¢ ´?pMZ)0[J>$\yen$??¦25?W2f@Ì}G@H?Àù9¡0iÃÃÂ"7?Dm($\times$1¨? ↩
    I¢???éE?2ÍÓÒÒ?t>1$^3$fÉ?è+ 8é’??ÓT?çß"pdÂÎÝÌ£Ã\Z$¡9(§(á!=:[5ùIYR$\yen$ÍiXÙYg ↩
    ??&3k3V0oQBÞc??d>"5*pæ
=kå\)-;¸é©$\times$,
N§Iia?öêBÈh$^2$69 ?+Kphþöy&??ÓezÑiÚ%wR[4$\div$å?úw¦0¸þVSBÑ¯v®êYÿÕMí^?Ë¸ÏR?6Â5??? ↩
    ËøÑ?Q,ülá¯þ?Úó[Û·´5ÑcÎ·?j\ensuremath{\pm}T¢/Ó?ôë$\div$Ó¼
ôîúWü·´U3\ensuremath{\pm}$\yen$ê,?N¤YeKæîl~ÕA??ã?oüy+ý$\yen$ÿì¿þ:àúËî¸¯\ensuremath ↩
    {\pm}ì0æ\tÊ??Õ^ÞêÍ,ß$^3$ôõr¯$\yen$J®\ensuremath{\pm}?çÍ$\yen$\ensuremath{\pm}$ ↩
    ^2$\textdegree{}$^2$àY$\div$$\times$ë$^2$à~/*ËÍ\textdegree{}¨ÜWëÎÈ$\yen$Yv$\ ↩
    times$¯}{õ/ù¸^¿üò1Ëx^$^2$õÿÍzÿ§Òàð¸¾?Âþ>ÿÁ$\div$?ÿ/oÅôàÚq¯=ü;Ï\ensuremath{\pm} ↩
```

```
 ¸ÑÊV-««Úíÿ&çóù\
ï?Ëõú?¯þyxu\textdegree{}5?û¿i?Íßí$\div$uÕuó3ÇU#Üù»?¯_îz|¯sÜù½7Çôñ¾®e¯?û?
ó¯}uÍÈèá}xßUÅå§'ó[ü?oñËøxüo_ðúÜ/ã??ëýÎU¶?,&¯ß ñ'ÖK04´$^2$s)ÖLJ)<®Pú9YùÚJm\õ ↩
    -]]-?=^Ö·o»ßî7?ï¿òìx??$\times$ù<û??7Ðó|ßCÏó7ÞF$\div$ya$\mathrm{\mu}$ÙìhV-T??Ì¸ ↩
    }~¿Wë´¶äò~?íÂþ|?Iÿýoàåó-°ì\F8Tõ?¾¯k¶ñ$\div$ë?7Ëà?]ûl¿ù$\times$ý¼Û?Ñ¾ñì,64ªeL)»L ↩
    .Ï_£iÉø>?/$\mathrm{\mu}$êë?¿ôö=Þ?½î{üoÏê·úú=?külL$^2$\ensuremath{\pm}2YGB?!s??% ↩
    X$^2$adÌÌÌí?FÇg·Ýî7;ï½??åØØùã?~ßÛeû?gÝ=?$\times$Àü¿ÆÇò$\div$s$\mathrm{\mu} $ ¨ $ ↩
    \times$ÒÒPêæU#HQ?óQ'å»w???ùäå3dÓ1«F\ensuremath{\lnot}ò8mÝÛL$^3$?9}?\ensuremath ↩
    {\pm}¢'P£P¦]BÉTìí%EEVÆ¦¢®$^2$¶®\textdegree{}~¯ÙÖUSÓRÑQÑÒRÑÍË©DùÒðà½pÉC??1\ ↩
    ensuremath{\pm}0û]çNúû$^3$Õ»¸¶¸èôú·W?ÿ] Ýíõýð]ÑVfS?M?æf¸zú?@?Ñ
Mì?O,\$\mathrm{\mu}$b$\yen$2ëf?,NBuIåe?"DxéÃ?2§n\7m?Ë?æ°|ùëfã®$^1$õó§PcE-~þ?Q
1aLCg?HJÙutTT4°úmãq¸Ûìö¶?Öï{»Ýïw»Ýàÿß?ñåÝ?[1ÕmmeufÄ ö$^2$·cWUSQSOOOKKE=7=76$\ ↩
    mathrm{\mu}$dªtRN?.J?.^ ÀbèN32$^2$$^2$2$^2$X?Ù?0{7Ý>¯føEÕçO«ÕéÞ^tîï/zw··$\ ↩
    times$·gÙöuz$\div$ÝûëÛûîÖ??&6?.?C?YlÆÐf5?Llõì?Ð_¿%$^2$ôRÑc?l?N?*!??9ÉWÑa(¢õåÀc4 ↩
    ,Æh&hü¡éD?å?^«O*????Q$\yen$N\°å«¦ggfæ&?-Z¶]q4zoT°fatÜììôôìüôììüý


??üôüüìüÞ¯WD<´°ú?´Õ55"?êêij)zmb$\yen$ RÊ¤>pn???¡ñu?%âÁvé»P8l>Ï^ï£õÚYòù53ò||_íí{? ↩
    ËùúÞ¿óþ_ÃÓþ>§ñþ?ÃÔõ¸?Ëùúþ¿$^3$í{ÿ¿ð|\W3iÐét°W]kî$\times$k$\mathrm{\mu}$Ç/39 ↩
    ËÇp2hú5#$LNÐÑÓRÔlv55uu»1·uÕ»=¯oÆ!?s\ensuremath{\pm}«©$\ÔSÔìjjuô´°ú*M^®~WE;;1. ↩
     b$\mathrm{\mu}$Ó?32ëV?KÌ-VVRQ?¢?r¸é2p°?B?è
4lâ1???ø
??ÓÑ?
s?åd\ensuremath{\pm}ÉaÙëÞôþËk_¤f?OÇÃ$\div$?·öþ~¿?$\times$õ¸>§ðô½?öô¿?ùzþß$^1$ýø?? ↩
    ÉäòùM§$\times$oqqwÓéôï»8XD?\ensuremath{\pm}ÌÍ?;?0O?Ô?RÓÕOÏë(5°Úmn°a\textdegree ↩
    {}ØSì5ÚÚMn°[\ensuremath{\lnot}~?.\$^2$YX*$^2$$\mathrm{\mu}$ë?K®Õj¦?  Ê??1°"VÖ«X ↩
    \ensuremath{\pm}BpAD¨Ó£?»)(?´?Äá¡à??+?h
ÑÂ? ÕÛý?Ü¶?>hÕ®kvà«y$\yen$·OA???:}? á?$(qÃË? ~;cl?0p0\\textdegree{}$\div$âÆ?H$\ ↩
    yen$??ðbÅ?°ráÛàC'MÛ4hÃ!Gs?$^1$Ù¿¿ÂÃÄÄÄÆÄÄÂëßàädbaàáacw?d
2d0SLæî????ò3?ê4tê???aÈ?N D?ò?Cu?J?'??"ta?X\ensuremath{\pm}F*?£ »?Énå$^3$w.(*ô1âÅ0 ↩
    ?hy?¢D£QÕ+V5/T+Cª$jèõb¢¢¢ÚúA$?úz}}-???ôþ\ensuremath{\lnot}J'Uì+PQ??\ensuremath ↩
    {\lnot}ÙÀU@L¤ÖÏ\¡f6?ø
?Ìef$^1$.¶Ì%k5fÕ~ó3%o»G/Ü?VGïÈLñà??øR?m+wN?ÃdÅL\°x8Ãä©«V3ÌláãÍZç9zð!'õóÇÀKG?F
?"|Ä=£H??~JJyTêU\ensuremath{\lnot}Z3¼°$\mathrm{\mu}$J?)XRÜ?*P?*©rå(N?:?.i\ ↩
    textdegree{}\\textdegree{}pRæP~(?2??Z1£?eÜ8DX½Ü??\ÅÍ.ì$^3$0á»mÅL$\times$mÜ7~Õó\ ↩
    ÇN3]6|î?d'å?lSP¸$\times$~°$^2$EH?¡O).ubT@§©?%C*$^2$m:A¼ÇLÊ?BhX\textdegree{}êE? ↩
    MC?|ùãq??Í$^3$3ç?hFA|*x0ìcÑ¨U5®«?IMWá$\times$?¯?Ä«®Ûn6Þ?Æ¢·cY¯§$\ ↩
    times$ÓÒQÑjèus¤´¢ñè?hé\textdegree{}TPj7ï\·j'Û)?l,?¿_$\mathrm{\mu}$+JôÛÛÚóy|~7 ↩
    ÉÊù9\¯'§Ì$^3$æý?U $\mathrm{\mu}$ÇLX?$\div$$\mathrm{\mu}$$\times$¿ÂÂÄa?ÅjM?â?à-? ↩
    $^3$ªeÅýgèõõ?UÛ[
¾ßtKP?¯#Èßïì|Ï7Ì\ensuremath{\pm}\ensuremath{\pm}ò\ensuremath{\lnot}GþÿÈ'©¼Ýï<}Öãm_ ↩
    ][[V+?C=30$\mathrm{\mu}$P´è°¨HJf$^1$Y?1¦?Îu.®.:?vöGÑgggÌùù_??Å$\div$ýÿ{ßâ{Þ$\ ↩
    div$½Äâq¸?#òòþ£imsÔëv;]¼,<\\vM¤Ü8rýëâZL¡7"tAKÄ?Ê#K+*??-]177;?«¢¤$\ ↩
    yen$¦¨ØÕìjB®¯¯¯Úí\ensuremath{\lnot},,?Í~ñx?þ?Á.ßãøãÿuãîü}ÎãòàÈÂí°úZ"Z?ai)K-  1" ↩
    kgD??@?ó?¡H:gÚ¿½¼»»°èý\Û>_+ä?Ü~7?áøxß?ÿøþ?ð|?oòr¾oþ9¿WÕõ}vÝ?¾ÿo?YQA"Ïõ?Bã¼ ? ~ ↩
    ©"T©R(X¸u1359>KÐÑÒ Ö?q?¨5zÊ?t'?YË$\yen$Ô¨N$Ü}\ensuremath{\pm}Û('ª???f<JI
A´?2?¿.JU<p
ÿ4\\ensuremath{\pm}r_0Á¡ê!òQp$\ensuremath{\lnot}¨@©R/ÃB%åÄ¼ [.©J¢hÄÂá0ãÂ=UA? ↩
    bl0Bði1ã£ðÒÃfÅ]cC?_?Ö !6?°'Y¤¢Àü4ë¸Ã$^1$?Äí_^õoo{?$\times$½kËÊÎ$\yen$Ñ. ↩
    ïuuokiÍæÙó$^1$ÜÐu$?Ï´'w@é[Ü]Ðu¯:Ýl?Ð¼¨·Ó?M[8RÐ CG?É¢H¢\]Zff'T~TT?@$\yen$¯¯zm© ? ↩
    k$^3$®?o?ý$^3$&þUÖÖ?<TTTÔÔSSRÒQjÇSóÓdá+wR¨ÄW\
Èx'KÁ¡®Å:$^3$ss?@\yYLX$^2$dÇ# a\textdegree{}6?,?Ï(~x~Ñ;@?ÏbØn]?)? Ää¡?C¯?IÉ$^2$éCU ↩
    ?Í~B?ÆO?~y7T ¢?*pÑb\ì!?(ò?ò?znÙ\textdegree{}®{A\?òÅøqñ\textdegree{}ð\textdegree ↩
    {}¶{WÄ?åõÿk$^3$$\times$êôúwW}> ï^üz=~¨ð^ÞõúãÚÂÄÃÃÃÆa1+@!p/Öî? ?Á?hdE? ↩
    ¸ÑAPqÒDâýdðH \ensuremath{\lnot}Dhì?$\mathrm{\mu}$?éþ?uª?P?Ù.HO OñÈ??|]P?øñ? ↩
    úcîù\ensuremath{\pm}ûIñG~¸*T!õZÂ'ÉX?2h¤Æ  $j?ø????
?p?i\ensuremath{\lnot}DEÝ<%??%>Þ
```

```
vpÙ\ensuremath{\pm}2ëri~a6‘[©:?$^1$È?¼%ü2\textdegree{}ÄÂÄÆîw10ÉÚ5??ÄÆÆ‘Ã¸’ô??|a½~? ↩
   r?>&6mKå?R?âûDQIøÁNãúDì Ý$^1$Ëá?$^3$?§G$\mathrm{\mu}$Ó¸þ ???¡??Û?Ýà ,t_Î? ↩
   wIçJDÆT?G?-[3551579<:??Ô4#$^3$Y\ensuremath{\lnot}?´üqüÚHþÕ

? ?è(?Í’Õ’Ú?ª&ÃL
?
F$?OYYyq¢?*?v???ïAY2EØSè\"\ensuremath{\pm}p M¨0E??ö#n??àA<?’?cÆ\?-??ÃâÇò= ???’Ó&? ↩
   oIæ|ú??$½?_?®Oÿ? ??l}§ä\ensuremath{\pm}pS=zðGÛùßvø´h@æ$^3$??DCm?Û6l?wÎ&Ä
xôaËd(êU?$\mathrm{\mu}$eÌgG\textdegree{}·bu?F?G¨à&ø?,ì{HQÒ&SSQû4ãÅ¯??ªÕÒÕ$\ ↩
   times$í«ë5ú$^1$Ú:»
Îâ¾®]ö®$^3$cMEGMOMD
?ª?X5Ù¨?eÑ IK¿¦kËæäÆ<t´?+ADLÖÜB$
|0vRU@6
à?YTiùée$\mathrm{\mu}$*N=~yR´¢´’äÁa?æ’:tÉ?¦ÕF??ZòæÞ$\times$Í?Aø\ensuremath{\lnot} ↩
   ¸ï?òý?|ët<,CsYmÀ?´ÀhÔ§¢L\ensuremath{\lnot}Îñ6J$^1$
IÿÃK¼Ýmötô563Ö?íæÝn(wt$\mathrm{\mu}$??:mÅH’¾$\div$UE)W«$\yen$KBî1Oï§¾Çq¾Lf?ú¿Ïìé¯å ↩
   /þ|Ïl!Aý$\times$úãÿÚ  kÿøßø^¿Òô¿ÿ$Û$\mathrm{\mu}$ø~Î2îÕ$^1$.2¡á#W
*Ú´$^3$ôTÕsÛZMî$\div$meaän¿u%åùÿ{eè?
k?-ät+¿%ÊN·ß¶ËÑûö^7£ú¼$\times$úýýåßÝOkkÔø¨$\mathrm{\mu}$ªO?IÜFg#bó
í½ýÝãKR$^1$}O©Ìæ/¸är?ü?#ÙâÚú¯º¿$\yen$ûøkþ~?cßä/ö9+OvÓåÚÚ^ül-$^1$·y<Ë«®ÆOs!Þã?mË?8 ↩
   Tc\ensuremath{\pm}??Pe-?ÓÜúI$^1$Ñ?4$^3$ukª’©u{ FÞ¦º§{$^1$ÞÔnì7?+Û
nïk$^1$Ûo·[;
Uøw{?
ö$\div$o$^1$Ûov$\mathrm{\mu}${ª?ZmÐ??\ensuremath{\lnot}[ZT§P?Ë‘\ensuremath{\lnot}$ ↩
   \times$Í2?d\textdegree{}¾û>Ï®ÞÖÛü}6vãìòy?W?Çø}¾?òáñÿ·$\div$öx¾ÿ$^1$Âàðx^çÃéü?? ↩
   Wë{¼N7?Èäüÿ?½ñZñ9Þ;¡Î´º$^1$$^1$»¸çßõ0È?NÖ.??\m?´|Þ?’ÏL>TS\textdegree{}M?d :a??
éÐ¢D\$^2$ÅjÖªÖMPÌÑÍPÎPÍÒk5´Ô44úï?oÃØUëª©¶Úh??Æ¾ªª§_Q¯\ensuremath{\lnot}$\ ↩
   times$ÔQÐÏÍÏÐ®]73+**CÈ?>‘\ensuremath{\pm}¸¦M? Äå>|S§á  H._9ê?)cèP?à%¡D?  /?L?%Á ↩
   ?ÁU?B7e%?$\yen$B?ç#Q  QÁ?%?¤H?HFP?:Eâ??¯ÈM’O?~Ä@Ï1\ensuremath{\pm},?Ã???7nû\ ↩
   ensuremath{\pm}}Ø¿ºiu¯?($\div$~?·?+{¿K$\yen$qsÒ$^1$’eÝÍÕÕÝåÝ$\times$S{Ø¿ív\ ↩
   ensuremath{\pm}ñX?TÚ4?ïp)ð??Õ4tüD¢gT«?{NO
Ák©ö?5Uu{*í¶ÚÀ~ÛKîH)MÖïÇÝûwºÜà$\div$[ Å?ÆÏk[YV?P¦!wgÀ75Sbãà£?~0@Ì#:?8X0øTv"_;??ÌhÈ ↩
   [\?O$[D&V: ?\textdegree{}ñg881Í¯Ç’$[©??Í«W?¢ØY¤´{Ç~op?]?(¢?Êb??F?ó2
Þ
ØJ#ÉÉÙIP&?$\mathrm{\mu}$G?ODô?ç?-Ã? ?$Ù$^2$6(\%#<PcÖ¨5j@V7pJ·6?{?Ú?$^3$%}bRÇ(g(?Ì0 ↩
   &?F"?hP *T~SÅ<?g?«?{ !À$\yen$V\M?Z»U0N¦f#´~?J??.1;àò(Ö??E+!JHe¡ª($\yen$? :zP?| ↩
   Ø?y-ÈJûE |£Ùõø??>ñÈ6¨sQ$\div$ÝgÒE©DCPTª$v6ÀØ6p?pày$^3$\ensuremath{\pm}Øí:lÔt?Â ↩
   <?Á.f)?Ý-????£È???Åá?î!Õ?æñ$\div$m$^3$P%?D?ð?pÿaÜ?$\yen$Ùìö&B?8$ë????õà?H¢"?þ ↩
   ?*??#ð
$^2$??45  ¤??n0?À @¶¸?â¡;j?â’n?ðv.?â\ensuremath{\pm}Y;$\yen$BUK?ll\textdegree{}?z$ ↩
   \mathrm{\mu}$cøløÕªiÓ?êQ©?ØIÉ¢??ÑÚ+?ôi  b#Y?Ù\textdegree{}ábA.$^2$£«^O4"j©DÑ2Þ~ ↩
   R*!&þ,r?~$^1$k?&q£ø©5þu~é?Âjd%<(Ø@#r7.
6F?7A£+??d?$^3$?¶?â?é??5ÀÏÊÉ?ö?ÚÈÈÈeÇ(?·’,Pã,v$^2$Ë ?ã$\yen$jÔTnÜ?eÓ¤DA?$\ ↩
   div$àT¤ÊÅEZ1?.XÀtGÂ??ÄtDÍx2 Ò?qÀÚpáe0?aè?\textdegree{}GÃ?¤M?4LC¼-à»DE|Mp§@ëîXÐ\ ↩
   ensuremath{\pm};¢Ñ
î?’\ensuremath{\lnot}Oã??À4É??Èn¨Dá¤qhr?U27¾|é?Ã$Â$\div$!?BÞGNåQÇ5~{?Á2ñúO*<ém? \ ↩
   textdegree{}A!¾~ª?;$^3$?ÂcDJ<ÐæØ?¤?Ð~ù?%1Høbp?~M ¼|ñüpÈ<u??¡?ñÆgàî$Çô|?Ñü®?? ↩
   ÑÔªøMCÇÏ¾á?.$
8BÛäåâ[ÈAD?ø¢¼?(??é¨è?????t?Å?K}Ðþb>ÁÇ,©’ÌyØ$px!äôut?ASâ@ ü?È(R?§8??w\ensuremath{\ ↩
   pm}BsÆO¦¡¦ØU$\times$x$\mathrm{\mu}$¶5õÖ?/]=UM»Ûnª¶uU$\mathrm{\mu}$»Ï:ÊËòØÏ+?yÛ ↩
   ??ñò½åëù?ô??Kâ$\div$ýÞ-î>?$\yen$ûX? ª¯¦O& JUQ?G?Ù¸ÅLÎWVë75~cÿ?J,½?þqô<ýþß}$^3 ↩
   $ÙÒKÒ¢??ß7?îÏíúöþ?ÿ4$^1$ü½Ïoê|{nÓ?;Î¿Óýúw¶5õ§jÚë«}ÕîfãBÍ,\textdegree{}ê¤$^3$´S$ ↩
   \mathrm{\mu}$?õ$\div$Èô|ÿÙû½?ÿZçýÖV^uã?ÿÇðÖÓTÏP¦G??V9}??¿Ò_êû?OßÂ$\div$}ïgÈãZÜÜ ↩
   \ô/úK[þÝÿc·ÖÈ½$^1$ì^öïz$\times$\ëÛ[K[»~mËîón®y¶cÅgÛg?%Ã¨??Fa*ÅÓ
wSa¸Ú??ü[
ç¼ÙxÕ¾-~ÒÃÅÙm7?O½´ñª¼]Þÿ}ø7Ë?&Çó~;ó~ÙÀüÞoéó}?Cæp?%ï{¼Üî¶uÔôz$\yen$ªÔ~8b?6\ ↩
   textdegree{}ðïn.º6’/íéÁï[Ûø?u?]Õö??&ð"Æ:ZéÝ}-m~ëwäù?Îô?öá?íô}?;üì|æóïxÕþ&ÂiZ4F? ↩
```

```
    õÃf\ensuremath{\lnot}rqq\textdegree{}:=?[>w=zó=5Ïßã{Ü/[$\times$õý¿cúÿ^?? ↩
    Ýá_éîð½¿ëý¯$^1$îûÞÿ½ïq¸¿?ÅÉø$^1$\gÑËú9ÖvG3ç??$^3$´çÙÙóímz?öý.ÅÕÍÝà?f??$^2$8$^2$ ↩
    #·ð-?@:~  Ùê?u@E«g$\mathrm{\mu}$"o$\mathrm{\mu}$û'fÜ?p"å¸Ü??uÕÛ'É=~Ö¿g¶ÛØm$\ ↩
    div$[}Åx$\div$w#Á·\textdegree{}ÚØlö{'-e]E-NÄ)?\ensuremath{\pm}$\times$ÓÔTì6?\ ↩
    textdegree{}]n$^2$?È?ÂydÔüÂÊS¦K&|þ1?V¸,ÐÅN>>.?íÏBÓô|¼7?ìz?$\div$õý_$\div$?wûÿó$ ↩
    ^2$¾§êúÞ·?Õõ§?Þ?E9¶Ý+ËÀÏ~6>7m£Æï?ÃdØ)??ih¡ÖRkv?$è$\mathrm{\mu}$ðêêªÃ!£
ëhh&¦¦¦V¿?ä*?*åQ©"F¢ß%?)FU>p¶
_
æ????©"la?Ejæ&çØ  ?n?ÿ<?? @êèèÁ©)?Áa)èíËÇ¦L"dâ«b¯âªb'Å?
??ë9«,¶?}»î$\mathrm{\mu}$çc?úþöû\ensuremath{\pm}Öê?J.¸X¡mÒ¸¸¶çÚóíú?Ö¼ûNw;èçÛ[[Û[ ↩
    t°WþÀN' >0H?'-i}\textdegree{}8?«CÇÑìV,Ö?t/8??Î ?òË?$\yen$,ÌÑ?°],$^2$YlÄÔÄBjh ?* ↩
    Dn?áÑ¤
Ö?{øú0$\yen$<v4$\yen$à?$^2$©e yy??>¢??J§ÐÓÔ$^2$¤S5?ágB{a$^2$?ìf Bñàã(dÅäÂ§)?!T¨X?2 ↩
    ¨)D??~?#???\"ÐÊ|ýá,+D??ü^GEà3ÿÍ?&í'_v?ÝÞu.îï?eííà??åö?u?8'ÛÇ?¶¤c^:¸ cdU<9Ø? ?H8 ↩
    ??F]bé\textdegree{} ý?¶aUáùëv[-umvÊ°»eâlÿ?jÊx@Jo?ôqRþr$\mathrm{\mu}$$^3$?EÜ+? ↩
    R3Y?~¯'ä#Då?´Ù?"~ ÂÓÑï.?$?F"?CôF5&'Ç?«aávÃ?]n·X?'$\yen$oÐ??Í?ÕËäòy???ñr>> ↩
    OËËåüüÎo<AMÝÝßRöÿ?·ÛÃÃÅÇ?ÉY?ËlÜÁ6\ensuremath{\lnot}3f???)[1?j  ú
?@$\mathrm{\mu}$F¸o)©ØS??$\mathrm{\mu}$¡'ª  ùÉÀK 6ÒI:Þ???¸,?ÒK(ü3XK@f?ZQsÃEÃ:qÆ ↩
    ?/,??$\div$Á~väU  \Í?b"\ensuremath{\lnot}!Uâ91??¸'h?á??????$\div$/cèd??à?\ ↩
    ensuremath{\lnot}u[?jqCÛ?LrK?
Á"ñË#Ùät~þäHätÂÃ¿30?xæi?$vDP?;iDm38sr?åÀà¤ ù??0þAaýÄL>'þ?ÌHìvÖ'?W?¼?"$?¤2;$\div$? ↩
    ÐGtr8EF?z?#?~?È.ÃädI?9?Îß¼ âGVÄéK?~?¤?á=?$*jÉQLÜÙò?<N?¢A??6?\ensuremath{\lnot}? ↩
    WË&\ensuremath{\lnot}53".?Î?À]YÙ1á?|??pü¨OÄ\ensuremath{\pm}À?rb1qq?q¶=pý%]iÛJ?Ë ↩
    <juÒuvVJW¯ë@_?ôOFÉ°ö1Ñ¸n¿
ÊÊY!¤J'Ù}?? go{Î?Ãä]N?e§ÿ?  \ensuremath{\lnot};½àY?ô$\times$ñú$\div$ðr¸wê%?ÄÕS.; ↩
    ½¦ö¡ý_$\times$ëêæð,\ensuremath{\lnot}¼¶?$\times$án\ensuremath{\lnot}¼°wn<Û+/Ñ< ↩
    wÍ§:vÇ}??êJ?¸Üô_Ë}É_ÁcâØ?)/$^2$¾Ñ<2rùÄ\ensuremath{\pm}/mÌþÅaúy;?ÌüíöÛh?$\div$W$ ↩
    \div$òùä ÍNÒ>
àO(ÞY.ôq?âáòø?rK;¾?á?Â??"yÏÂõyý$\yen$s\ensuremath{\pm}íNØXÙ?ô´É¿\Lõ¾ð%qÿA§PiéVVI ↩
    ?1FvÆ?wk:rÉ^$^3$l8f?géßþ{þOK;Ù¿ä3ÃIZ$^3$é6Ã{=½,?«ÕI$^3$ý^e0à¿¢óÜÅ4{ÈÉ}«_5Ùk¼$\ ↩
    yen$ÿWÁoÙÛB»^¿$\times$ãüñr/VÝ[Òí\°E9¶=9Wú¼ï?ñY?NÊÈ}ú·ù?7eäýåÔÝÞa¾+òrXõ®nr=;¸ü# ↩
    ¾Òþ??$\yen$êYú|f3TÎ?\]?âsu¸Þn·ÿÅçÿÇ?ü[Ï8r_Ëçp??üõJ?iãÒÎÏÆ'Ü$\mathrm{\mu}$côdû+ ↩
    øÜZá°¾Ïå&r_Â¶$^1$sØÀÆ??Áä54%ÂñjãàÚÛ?ùÃ>Oå$^2$ÿ¾óÄ¯Úm??s$\yen$i$^3$¿Åë¶Âúÿè? ↩
    ÜïÑ_Ãã¶ÆqYM)ªS)<aÒ
?Î?jÞïö«j<~)®?y?Ù[L¦x£Ê&VIÞólÅépzÿsomoïx8?Ñ{ì/_òðù<?ÍË??§iY$^2$©"è
Ox¶6V;°ÿ?Ê&$\times$kk}?Í¾K>JÚzíô´5shÙ½\äW]t 8|^ÇK¿$\div$¯âñ.?$\mathrm{\mu}$ìZv-ù\ ↩
    nÃFE|þç©gÚËT$^3$[WZ°?þs$\yen$;¿´\textdegree{}ªÖÐÖ$\times$ù>âPIÉ«Öï>ýj"~3
"ÙÃ$\times$\®£gñrð-x^?¦½.õ»Ü[_M©ñYÜä?P4uM?'_~V ,V§ut$\mathrm{\mu}$Õ?3úý} ↩
    JéCêè¦TJÍÊÃÌ;72õ½ß7ôýXÙ¼¯gûý6·vvÖ$\times$\ÛK^/s~Åbï·Ún
?-?j'5Þ?Jä??&bcÆÝU+2XÉõ)T#8mZÃq?Å.û?þòöÞêþ$\div$$\yen$ôZq>{kÜ.Þóû?l?
Ü¼*?gÐ?§Hf?c¨Ó¢N$^2$r¦d\Xk]®N
Ó?
Í<?Ì?!,õÀ£?ÌÎÕæ?c?Øw6¸ùyxí??[8?Lì$^2$îÎ¢LªN?N??Êfêé?¿tå¼TÊM·ËÇm??v®$\mathrm{\mu}$$ ↩
    \mathrm{\mu}$¯?Ü$\div$z?Lû|[Ì´çt°½?\ensuremath{\lnot}?[£oxÅëpØ7O$^3$èê5*£> ↩
    Fª¿gYW]S¯$\yen$§¦ª¨TòÊjiù
Ó.}"@?èoEÝÚ0¾êsùüÛ[®Ö^?Å$\mathrm{\mu}$·?è·»ê]bããà?´$^3$Æ¾i1nQsÓ#7&A?ÇJÑäS[???,?OM¨ ↩
    \ZS/^5föOS$\yen$ë,¶Û$^3$Æi]KïnyÜÐ??ëûÎ$\yen$ó0ëè¦@JCÈ? -?,õ&¶na2i?©[\ ↩
    ensuremath{\lnot}ÖÐÏÏK?øÚ?Ê?$Bf?øÇP?Ï?ÿ?©E!@aóVï^?4BCðßÃ:R3ÉÕÌ«[I\textdegree{} ↩
    ¤ÖÕSMª¡¦§§¤V$\yen$:ê??È¡À.'¨Ëº Ìíýw?ä(%æ?%õ$\mathrm{\mu}$¯F$\div$?£,\ensuremath ↩
    {\pm}\ensuremath{\pm}óÄÛ·þ?lb?eñì',Tä2?W?A76¤ésê&è?)J|\textdegree{}:ÔÊQ?uÖ?hp@$ ↩
    ^3$ó3$^3$ËtQ@ÿ\textdegree{}
¦V$\mathrm{\mu}$*?4ÇÏ&~[&D0?($\times$Të§?<s]C8$^1$H?¢Ec¤P$^1$Pc?P¡?fì7?¨EK@Lá? ↩
    ËE1IC0©:$^1$¡?Õu¾?$\mathrm{\mu}$rTk'õ$^3$éÑ BbY2UªÑ?.ò?¼agÌqo$^1$¶·=¶Ïa^Ýu\ ↩
    ensuremath{\pm}Ý=sò?3'?jL@R$\mathrm{\mu}$´4$^3$ë?LO?j(§èê6?ssÓóÔ3sj??Ø£R6\ ↩
    textdegree{}\textdegree{}CÎ?â~¾?¨nÙìh/\À'¨§*JØGÌ-X¡ Ô
&'&$\yen$ÍPÐÑì*uÓÊ??¨Q¦XÌ?9ÙÏ¢,ýÃçî$^3$?ÅB¢LÉPËDZ?t´éÄKV&?BD}"É?(???\ensuremath{\ ↩
    pm}??´cdäÒ~*?$^3$ÈNÆ?ÑN\<?}z3Ð?  ´¡=dÊ?~ >?Ã¡§ª«Ùl¼:J?êz°}mNÓiUG2@*ÎO.J,Dy?ãA. ↩
    RÐ?N½eÜf?:oÝü?9\ensuremath{\lnot}q¸?1Ã0B|Fdå¦u$\yen$$?*YrÁ<?Ó.R?Ê?Ø¡"ªQ???0?Ì0? ↩
    £?Çð31p?9gÐ$^1$?«ñ¡¿09?0¡Wc~TJ?£L~$?§ÔË,NÑNF@9ÓeâÂXÑ?a(ó
j)ëÈEmÈBW¨%K?[Í*?c5\ensuremath{\lnot}$\times$ki')?#=0?w7?4¤-æ?i3\textdegree{}ßRâA| ↩
    à?iØ8eË9sçA?6lÛ¿??¤$^3$È??.xábçL??J4  ðxáôÔÓ??7{
1$^2$Îr1Á$\times$Û3jñãfy8øy??òò??Èöí
?4xÔX~$^2$hM  ¨!J!âðâ¡?âJaübã^8
à?Äh\textdegree{}_¼~ø|¶lñóflÛ8r?GÆ?? :b+æà5ðã?nàJÐ@Á'Dü8ÝÑ$^3$æd ?7cÖÀcÝo©uØÅdá» ↩
    ,\8öm2òrBùl1Ù3zå\ensuremath{\pm}©!?7?éhfÍ6m  %A¤Ò¤H??!fÂ&b?Z?82DhÐ?@+ù??1©\ ↩
    textdegree{}ÍD?z,f?Âå$1¼($\yen$Ï!JvLñÄ?g)?M?Ñ??
?fÏ
(Áp¢??·zùÌ?^õÎs¢3
?®[(p¢?éAó?Z?ír$\times$1Ît7E??b&Âq?à0Ð¸ûf-?å\textdegree{}?ßlÕ?Ülf9?cád?ðîÏ??ìÉÃÐ;· ↩
    }?$^1$?Ð¨i~qðK?j
c?&Èá}û$\div$NK¦ÿÒdIl$Ù?¤à?â?p?x??"4?(*Ul$\mathrm{\mu}$·ªúî¶õ«$\times$kmnï©7{ ↩
    ní¾o5Úë[ZÛÕå*ÕmZ5j$\mathrm{\mu}$?(T8(?@??'~t2C¤É'IM'$\div$ûþÿã½}}Ï~/{3ßÙïfoæ$\ ↩
    div$Ù~?vîÛk#i(?~YR$\yen$)è$@zú? >?Dg? 5?n$^3$$^3$»ºAA^']$\times$
ûº0cE~??0?(?40pt¨\ensuremath{\pm}%??ct???.?$\div$ï?>6iãOà=x'Ñè{=ôIGG?T?$^2$<+Ø? , ↩
    D8?á(Ó? T(F?¸b$\mathrm{\mu}$ ÂTT«N?
TéÑ?&<H§~L$å?¢#À22ø2DàîªÄ4®bÄß[10?b?ò¡"\textdegree{}?Ád?Z?lldS?Ð¡?\textdegree{}% ↩
    ÁÂà>??Eøötç?
á'P?]
??Vt¡2ò$~F<?ká4'¿¡§0G?zZ+???n?'Q¸âÊ?]¢?A?é?$¡UàäÚU
V-?ðBN¡(Dr9q¤eD?RL»È»0 E" ?$^1$Ç[P@'½P\ensuremath{\pm}ø>RÅÓ??Ô$^3$?Z@|àT
Ô?øÕ«?<?$^3$FlÔAÀåI%õ#  0?p?À$ô(?¾ñ"tJø&*P?ó?Ý\textdegree{}?er$\mathrm{\mu}$*W0.8K ↩
    ?,Xh?~$\mathrm{\mu}$)Bøáw¤e¯?C?HÔ¦héBåiÔ)V?m?P " 3 >Ét2å4Eì''V?´ÈP£JfBr¤?D??F??? ↩
    FF6DðJeQ$Eø?DËO^?ub 3Y?K?J,Ìb?Âé2pKz?29#&22hG\textdegree{}q?$\yen$??@ÉÉFÇ¨JL´
Ð9?Tró!h4$^3$ 6a?æ9HÜ?Rý$^1$Awø~KQÊ?S"*K?&6Ç??$^2$Ý¨É?'jÔDjÀ?Ìh?c(®$ËK???rPä¯a0Uß ↩
    \\ensuremath{\pm}a1x¾aÂåJ?MÅ&B*#aÂ+%¨ª
ÂØêÔ§?4*Ø j¤Á}R?C?c9*E??
qBMC
l@Nû7?=#I?¸èá$^1$?C,b?Ié?'%1N?ÔLÝ?#àY?2É1!?F?3Eñ?Û£?ÄÎt¢Ò?»ÎO?#bDáðA$\mathrm{\mu}$ ↩
    ~(Õ?q@??àÅ$S?ã®f
©C?7$++ìy??i?.?òQ£?!?(©?5\$^3$?Y'Ð?IGÈ?DÆ/»*D´*FäÅ?TK'1Ä$?¤?í"?Èä?Ì?ÖJ#?t6Ðô ?jWò~ ↩
    ÖUB???\¨Ëä}ÌêY}Bpùp?t´tËà?y¼.Ç?Ye$\div$Dz¤&Ä%(bGK??Ô
"583()?TX$$\times$ÍË¶?$\yen$
SLÊQ?í8á¡Ñ?(ÆÊ?7@oXpso?k9D~Óã7ÄZRQ4MÄS?Þ<X¤çDc4è$L???$uÃbE6ÑÇ'BñdáÌ¢"¡?ø?Õ¤ß¢®?ÄÞé ↩
    %Ôyh¾b,¨8Y0B$^1$#
Ì%G-ÓâoÔgu  Lj¡§J?§LQÀ?º-?*Ú???HBHå¢Èè
ãE£
?4p=g$\mathrm{\mu}$?<l©@R????øùp?É??Q2F&å?ðr}î?¸ ~½ð,rÿì?Æ"ÒOk{Ã%?$^3$cI#4ã&î4Üòð ↩
    ?ÀÞ?ö?RÅ¡??ÅJQ¡3yé¡m $\times$?LX%y  Éùéa&§¤üä:  \$\yen$Û?+O?êÈz'Äýt?Wì^â¯ÒÄ¤Ç
r$\div$9dÌç©Ð&?%mjÜü?0? G)û=_Õ8É\ensuremath{\pm}ÐÕ¸Ê§?iâýì;?$\div$?ðWéÐ]ÐbÉÛ¦? ↩
    ùuVòÂ?8?öMÏ¦?t$\div$«1Ïw[¤æ?¢4Ì.?'M3ÝºðâZ xöW?'¢Uß¶9ìCÓ=øº(p|?~v
Øº\ensuremath{\pm}Ø|¶G¡qU?ê8$^3$-ý¾?Rs¡£Óh»G?æÕ/W½R¸ã$\times$ðù?Ù:Ö}ZP?z$\yen$Rçò ↩
    ?<]?o¾?lc$\times$Õ¢:jõÑïg?{4s©$\div$Gé?¸åOX¢xÜè1*Ð?C?ïOUv?ÖváZãðL¿ò!ñØ?$\div$IX ↩
    ?+Ý'ç?4l5ÌÌºdzWëÖR9vÅz?\ensuremath{\lnot}q?Ìg!É©?c9?)ê¤ÄtöFbå1~çP"w[¦?ku!ÂÃÝ$\ ↩
    div$9?ç¯ß\ensuremath{\lnot}ÇQ$\div$¼FçI;·h_ÉJªRm#¤§||:$\mathrm{\mu}$\textdegree ↩
    {}??>?'[?@<àû¢Ï¾K?D?Od )Â ÔJBñL"?ñç*Áû()?c?ÅJd*?£9J$oÒD*Ý?$??DIÑ¦?0¡"~B?ðÑ|rB$ ↩
    ^2$ M?ã#¼?xôj-RõH$\times$$\yen$?"'$\yen$$I??~üÑÔÉE(Te1åW?0%?E!Õ£9P!Â$^2$?Þ?ü*Á" ↩
    P?òaÁ?#?Ê=  3h/g?AÑã¡Iþ?I,?1@ É?è\textdegree{}bÀ¤$,m?@!dI'ÃB<?E4E?EA?)$\times$ ↩
    =???H0Z3$\yen$âº?Î?X???\textdegree{}®???W!2p_\ensuremath{\pm}Ci%Z@¨^? ↩
    ÜdMx8Ç0VºaRÐÏÕ Ö#N??P8¤?¡Q?M?K??V'IS¢??B?F ZÕ??Âó#ï(?S !ý¦?Z'ÙR&R?JBG?
¨LKf¦R=ðÑø!ÈôÅù?õ?,?\?D?ú?¨\Tä~>\aZdã?*DRuJ??$\yen$
$^1$??äb£*?F?D5?@??íD*\textdegree{}{I'@ZG¢?øZ=$^1$uZRÕh? ?àeô??Ùa, Á?zå1??$\ ↩
    yen$Rb¸Â\ensuremath{\pm}ÁS©! (?\textdegree{}V$kB'?à2ä?w?¢ÐA3©?C~??18Ø\ ↩
    textdegree{}i?}p?ñ@-fR\yQ9:?8\ensuremath{\lnot}«Â?ö\ensuremath{\lnot}«R?&?MÄ  ? ↩
    BÆ¿«-sT?¶;
```

```
7 ë¨+Ü"L§½{äI?áÆ??k?D|?á?
?$\mathrm{\mu}$èÀ ??ÒßQ§)KQ?V´??<£7Ù J?
ó$\div$rì$\yen$ ¢\ensuremath{\lnot}WØFXq$\yen$F?\ensuremath{\lnot}?\4 ?XöQÒ??w 6vv ↩
   ?’#F&Ç¢Efw D" zÍfi$:\textdegree{}??fQxA?Èe¢,ÀåB??%] àxt?©Ç?<É#~ÉÏ ;J?‘]Ø
Ü?Í?^Ü$\times$ô+ÍÈ>QNÐ$^1$a9?qÅnÉ?;0Ç?edBD?4Ò?f?z~E??á&Ø[Ô3É?{?yLÑ¡4Y??j??|  ↩
   ÌÓçÐ®$Ó»$\times$Ú!©Rü$\div$.]«@ÕødÑ{àn? ÑCRpÓøÕA¦?A#E AE ÝDñb?ÃSñ!Æ‘gF&££??þ  ↩
   F@é%õß&Ñ¤Ê8ÂV??G¿?q$U??- ? 6$^2$9~A´80?móòÍ0Åâ ?¶èQ$\div$pJü\ensuremath{\pm}b ↩
   ?\?1?fb@PoIS?´Ð6J_&?RÔ2ï3j¯Ñ?TÞH’¨qÌÛÇ?)R?cãËtlq ?P,??ò"Ã?V?AÄ?Øâ
¤GG?PbCKZ:fêFAO??áN???Àã\textdegree{}ÕPjÑ
§ÄÞ sq©$\yen$Æo?·x¢l$^2$?H??ä=I?°C???Ç \J0u?B?JCH??Äq?Í?þ?ùA,cD"ÀZ?-S?äÆ8Â¦?§!¢?¯Ì ↩
   ~R$^2$ï=?jB´*rË?\¾ÂDéÂ?¼@$á?\textdegree{}"?(?D1ò?P?Q¨RTè$\yen$D?ÄÂ<Å¿?Û$ñO.<??©
È¤J0øí?\J\¨;ú
l1?¶h arranged alphabetically by
command or topic.
```