A Hands-On Report: Commodore's New 128 Computer

COMPUTE!

$2.95
June
1985
Issue 61
Vol. 7, No. 6
$3.50 Canada
02193
ISSN 0194-347X

The Leading Magazine Of Home, Educational, And Recreational Computing

Apple SpeedScript
A Powerful Word Processor
Complete And Ready-To-Run Program Inside

Webster Dines Out—
Action Game For Youngsters
Programs For Apple, TI, Commodore 64, VIC-20, Atari, IBM PC, PCjr

How To Buy The Right Printer
Which Features Do You Really Need?

Commodore 64 Disk Editor
Examine And Change Any Disk

IBM Variable Lister
A Valuable Programmer's Tool

Page Flipping On The Atari
Create Exciting Animation

Apple Universal INPUT
Making BASIC Better

June 1985 Vol. 7, No. 6

FEATURES

REVIEWS

COLUMNS AND DEPARTMENTS

THE JOURNAL

NOTE: See page 110 before typing in programs.

AP Apple, **Mac** Macintosh, **AT** Atari, **V** VIC-20, **64** Commodore 64, **+4** Commodore Plus/4, **16** Commodore 16, **P** PET/CBM, **TI** Texas Instruments, **PC** IBM PC, **PCjr** IBM PCjr, **CC** Radio Shack Color Computer. *General interest.

TOLL FREE Subscription Order Line
800-334-0868 (in NC 919-275-9809)

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies:
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019
Address all inquiries to:
P.O. Box 5406, Greensboro, NC 27403

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published monthly by COMPUTE! Publications, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408. Domestic Subscriptions: 12 issues, $24. Send subscription orders or change of address (P.O. form 3579) to **COMPUTE!** Magazine, P.O. Box 914, Farmingdale, NY 11737. Second class postage paid at Greensboro, NC 27403 and additional mailing offices.

ISSN 0194-357X.

How to avoid paying your bills.

by Alan Greenspan

Alan Greenspan, Famous Economic Advisor

"The other day, a prominent politician in the executive branch of our government phoned me up.

'Alan,' he said to me, 'the budget is a mess.'

'No joke,' I said.

'Not that budget,' the prominent politician continued. 'My budget. My checking's overdrawn. They're threatening to disconnect my phones. I even got into a shouting match with my wife when I tried to lay off the servants.'

'Civil?'

'Not very. And I think I'm about to be audited. What would I show them? Who keeps receipts for embassy parties?'

At this point, we were disconnected. And although it was too late to teach proper money management to this prominent politician, there is a lesson all of us can learn from his misfortune.

Everyone has to pay their bills, and nobody likes to do it.

You can keep file folders full of bills, drawers stuffed with grocery receipts, envelopes brimming with cancelled checks, and at the end of the month, it still takes hours to figure out just where your money has gone. Not to mention how long it takes to straighten things out at the end of the year.

Well, after years of financial consulting, I've discovered a way to avoid paying your bills: let an Apple® II Personal Computer pay them for you.

There are several advantages to letting an Apple handle your finances.

It will save you time.

It will organize everything.

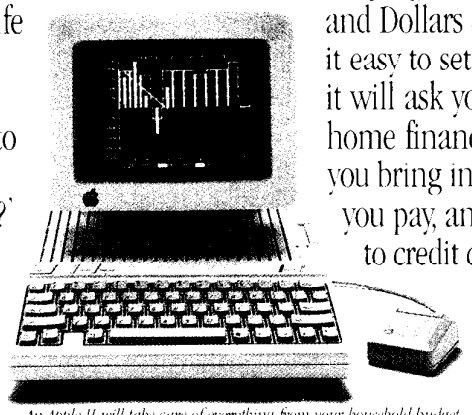
It will tell you, at a glance, exactly what is going on with your money.

It will pay your bills, and never send you any.

And now, I'd like to turn the page over to those nice people at Apple, who will explain, in their own excruciating detail, just what I'm talking about."

The Apple II and the Home Budget.

With software programs like The Home Accountant™ and Dollars & Sense,™ the Apple II makes it easy to set up household books. First, it will ask you some questions about your home finances. Like how much money you bring in each month, how much rent you pay, and whether you owe money to credit card companies, mortgage holders, or any other surly characters. Then, it will ask you to enter some of the bills you receive each month whose prices may vary:



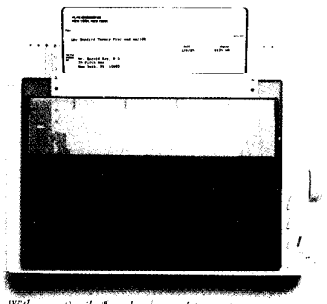
An Apple II will take care of everything from your household budget to your taxes with software programs like Dollars & Sense, The Home Accountant, and Tax Preparer.

phone, utilities, and the like. Then, it will ask you where you keep your money, and for the numbers of your various checking and savings accounts.

That's really all there is to it. After that, an Apple II can automatically write checks for all your fixed expenses each month. It will also tell you what other bills you can be expecting, and when you enter their costs, an Apple II will pay them, too.

An Apple II will see to it that your checkbooks remain balanced, and that you'll know when your expenses are about to exceed your income. It can even help you plan to buy a new car. Or a home.

Or a fur lined boat, if your budget permits.



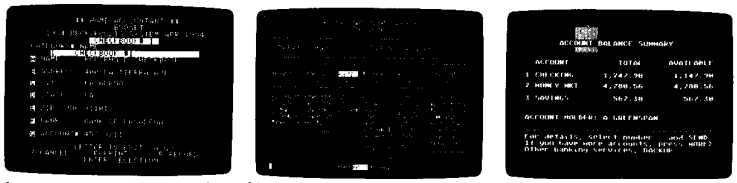
With our Scribe® color/graphics printer, you can automatically print out your own checks — not to mention reports, papers, almost anything. Except money.

How to avoid your banker.

After the Apple II writes your checks, it can call your bank with the help of your telephone and an Apple modem. And faster than a teller can say "Next window, please," you can find out all your balances, enter deposits, see what checks have cleared, transfer money from one account to another, and even pay off some of your credit cards and other bills electronically — without ever writing a check.

So the only time you'll have to go to the bank is when you want to visit with your money, personally.

Which, when done in moderation, we can recommend most highly.



*It can manage your entire stock portfolio with programs like Dow Jones Investor's Workshop™ and Charles Schwab and Company's The Equalizer™. It can even show you what's going on in your bank account.**

The Apple II and making money.

An Apple II can do wondrous things for your personal finances. With several different software programs, you can become your own stockbroker. Again, by

This is an Apple modem. Not much to look at, we admit, but it does let you pay bills and trade stocks by phone. It also connects your Apple II to a wealth of information services, like THE SOURCE℠ and CompuServe®.

using an Apple modem, you'll gain instant access to financial news sources like *The Wall Street Journal, Barrons*, and the Dow Jones News/Retrieval® service. Find out what they've been saying on *Wall $treet Week*. And in most cases, get up to the minute price quotes on over six thousand stocks, options, and other securities.

An Apple II lets you buy and sell securities right in your home or office, at the moment you want to make the trade. It automatically updates your portfolio and gives you detailed holding reports. It even produces charts and graphs, so you can quickly see how you and your investments are doing.

A little tax relief.

If you become perturbed everytime the subject of doing taxes comes up, an Apple II can do them for you with programs like Forecast™ and Tax Preparer.™

It can store your records, plan for the next year, and calculate your taxes.

You'll be alerted to payments you've made over the year that may be tax-deductible. It even keeps year-round records, automatically updating totals and making corrections for you. It will even print out completed tax forms that the I.R.S. will accept.

And it can do about 10,000 other things totally unrelated to taxes or this ad. So there's no telling how far an Apple II can take you.

"Well, I think that about covers it. And what if, after all of this, you still have some money left over?

Congratulations. You're doing a lot better than the government."

Solving Common Printer Problems

Selby Bateman, Features Editor

Few things in computing are as frustrating as a recalcitrant printer. Here are some tips on how to find relief.

At one time or another, every computer user looks at the paper rolling out of a printer and sees something that seems to have been sent from an alternate universe. *That's* not what I told my computer to print!

Your neatly formatted double-spaced letter is being printed all on one line. Or your beautiful four-color screen illustration is appearing on paper as a series of capital E's. The italics and underlining you've added for emphasis in a report have changed the rest of the words to an unknown foreign language. Or, perhaps most depressing, the paper is simply rolling out of your printer completely blank.

Nine times out of ten, your printer problems won't be mechanical in nature. More likely, they'll fall into one of two major areas, which we may call *interface/configuration* mistakes and *special effects* errors.

Problems with interfacing and configuring your computer and printer usually happen during your first attempts to connect everything together. But confusion over special effects—such as boldfacing, underlining, super- and subscripting, and graphics—can happen even to the most advanced computer user.

No matter what the cause of a printer problem, it is a frustrating experience. Yet, with some patience and a thorough understanding of how the computer, printer, software, and printer interface work together, you can unleash all of the power and high-quality performance packed into today's printers.

Whether your computer is a Commodore, IBM, Apple, Atari, TI, or other brand, you should become familiar with how it connects to a printer. Often an extra interface is required to allow an otherwise incompatible printer and computer talk to each other.

A thorough discussion of the many printer interfaces for microcomputers could fill a book. But basically, your data will be sent from computer to printer either in a *serial* or *parallel* stream, one bit at a time or eight bits at a time. Most printers use the parallel method. Your computer and printer manuals will tell you which kind of interface to use. But you should also know that some computers require additional accessories to work with certain printers.

For example, the Apple II needs an interface cable and either a parallel or serial interface card. IBM PCs need either the standard printer interface card for parallel connection or an asynchronous serial card. A Commodore 64 can hook directly to Commodore printers to make use of the special graphics symbols and reverse-video characters, but if you want to print special character sets, different type fonts, or foreign language characters, you'll need other printers and appropriate interfaces. Similarly, Atari computers hook up directly to Atari printers, but require the 850 Interface Module or a substitute to work with other printers.

That super-low-priced printer might not look like such a bargain when you arrive home and find that you not only need an additional $35 cable but also a $100 add-on interface. Although most stores selling printers have salespeople to help answer your questions, you should still do your homework with computer manuals, magazines, and books.

To add to the confusion, the application software you want to use—such as a word processor or graphics program—can add its own complications. Unless you configure your system correctly, what you end up with may be quite different from what you want.

For example, let's say your interface automatically sends a *linefeed* instruction which tells the printer to advance the paper. Your word processing program may already contain a similar command. And the printer, unless adjusted, may automatically add a linefeed as well. As a result, when you try to print out a single-spaced letter, the printer may be following instructions to put two or three linefeeds between each line of print. Conversely, you could also end up with no linefeeds at all. The entire letter might be printed on a single line.

The solution, of course, is to enable or disable the linefeeds, depending on the problem. This may involve opening up the printer or interface to flip a switch, or issuing the appropriate command with the word processor program. The answers are buried somewhere in the manuals.

Once you've got the printer and computer connected properly, you'll eventually want to take advantage of the advanced options which printers now offer. The special effects which turn your system into so much more than a typewriter are *theoretically* quite easy to control. The complexity stems, once again, from all the configuration possibilities. Versatility has a price.

Let's consider an example using the *SpeedScript 3.0* word processor recently published in COMPUTE! for Commodore, Atari, and Apple computers. To underline a word with

SpeedScript, you send a control code to the printer which backspaces and underlines after each character. But to Commodore 1525 or 801 printers, the code that most other printers understand as backspace is read as a command to enter graphics mode. Such conflicts are unavoidable, because there are so many different printers and control codes.

Whether you're just getting started with printers or are moving on to advanced printing features, there are a few basic concepts you should understand. If you're having printer problems, check this list to be sure you're familiar with each item. If you're not, invest some time exploring your computer, printer, software, and interface manuals to find a solution.

• **ASCII** (pronounced "AS-key"). American Standard Code for Information Interchange. A code that uses numbers from 0 to 127 to represent letters, numbers, punctuation symbols, and special control codes. Each code number consists of seven *bits* (binary digits). An eighth bit may be added for *parity* (see below). The first 32 ASCII numbers are control codes which can tell your printer to perform actions such as linefeeds, carriage returns, backspaces, and vertical and horizontal tabs.

Commodore and Atari computers use a slightly different form of ASCII which can cause translation problems with some interfaces and printers unless they are correctly configured.

• **Baud rate**. A measure of data transmission speeds, synonymous at lower speeds with bits per second. Computers can send data much faster than printers can produce images on paper. Consequently, the printer and interface must tell the computer to send data in bursts short enough for the printer to handle.

• **Buffer**. In a printer or interface, the memory area in which data is held after being sent from the computer. Printer buffers can be as small as one line of characters or range upward to thousands of bytes of data. If the buffer is large enough, it can hold all of the data you want to print, thus freeing the computer for other tasks while the printer goes about its work. The printer controls the speed at which the data leaves the buffer and is printed on the paper.

• **Centronics-standard parallel connection**. A printer interface which allows data to be sent along separate wires eight bits at a time in a parallel flow. Most printers use a parallel interface to receive data from the computer. Some computers, however, must transmit data through a *serial* interface (see *RS-232-standard serial connection*). The Centronics interface, named after the printer company which popularized it, is the most common type of parallel interface on personal computers.

• **Character set**. The letters, numbers, and symbols which a printer or computer can produce. Note that many computers can display characters which the printer cannot reproduce, and vice versa. Some printers are capable of printing foreign character sets when you change the DIP switch settings. (See *DIP switches*).

• **Control codes**. Nonprintable commands sent from the computer to a printer for special actions, such as backspacing, carriage returns, linefeeds, tabs, and margin settings.

• **DIP switches** (Dual In-line Package). Small switches located on a printer or interface which can control a variety of options, such as baud rates, automatic/manual linefeeds, printing impression levels, international character sets, types of paper, form lengths, line spacing, and other parameters. Some printers and interfaces allow easy access to DIP switches, while others require you to take apart the case.

• **Emulation**. In terms of printers, a mode of operation which mimics another type of operation. For instance, some printer interfaces let a non-Commodore printer emulate a Commodore printer, allowing you to print the computer's special graphics symbols and reverse-video characters.

• **Escape codes**. Control code sequences which let you print certain characters not included in ASCII codes, or which activate special printer features such as boldfacing, italics, expanded or condensed type, and so on. Escape sequences are preceded by the *escape character*, ASCII 27. These sequences can be sent to the printer in BASIC from your computer keyboard, or by the application software (such as a word processor). For example, ESC H might represent a British pound sign, ESC P might turn on or off the proportional spacing option on your printer, and ESC BS may determine the amount of space between characters in backspacing.

• **Firmware**. Software permanently burned into a ROM chip (Read Only Memory)—a cross between hardware and software. Printers contain firmware to control their printing options. Sometimes you can replace this chip with another to add more printing features.

• **Parity**. A way for your computer and printer to check the accuracy of the data being sent. An extra bit is added to the end of a seven-bit ASCII code representing a particular character. The computer checks the extra bit to verify that the data was not scrambled during transmission.

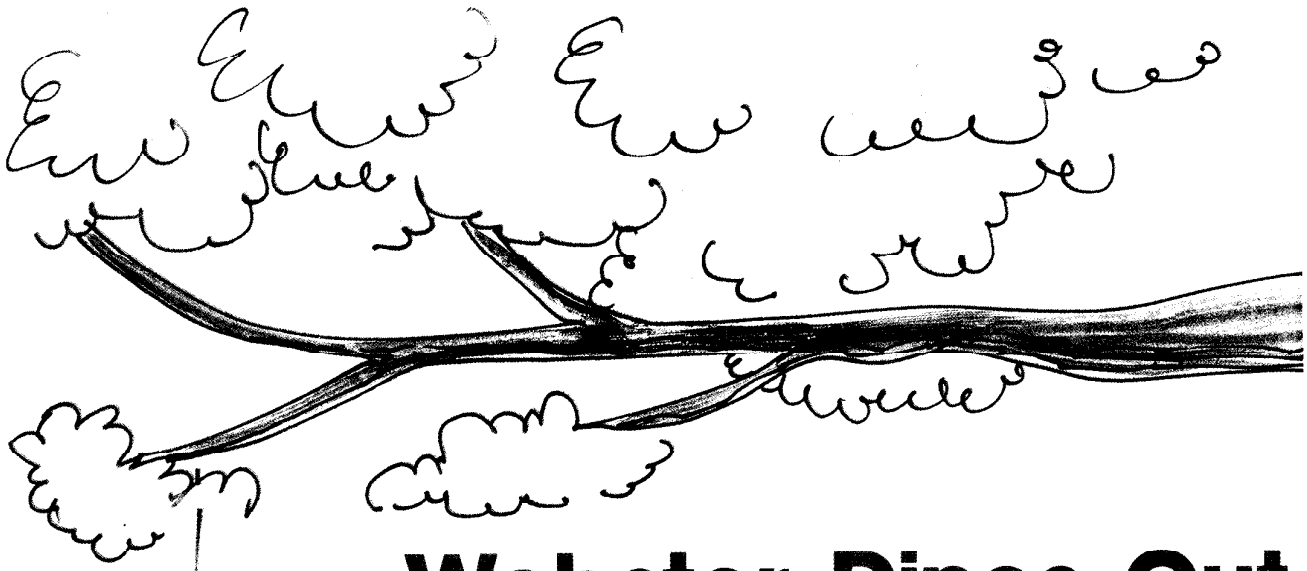
• **Proportional spacing**. Many printers today can vary the spacing between characters, as typesetters do. Typewriters have fixed spacing between all letters. For example, proportional spacing allows more room for a capital M or W and less room for a lowercase i or l.

• **Protocol**. All the rules and instructions controlling the way in which data is sent and received between the computer and printer.

• **RS-232-standard serial connection**. A type of interface that transmits data along a single wire one bit at a time, or serially. Although most printers use a parallel interface to receive data from a computer, some computers and printers require a serial interface. When all other factors are equal, a serial interface is slower than a parallel interface—but this is rarely important with printers, whose speeds are determined by mechanical limitations anyway.

• **Tractor-feed**. A pair of cogged wheels and guide wires that helps continuous-form paper roll through a printer. Some computers have built-in tractors, and others offer them as options.

• **Transparency**. A mode of operation for printer interfaces in which serial data is changed to parallel data without converting the original values of the data. ©



Webster Dines Out

Walter Bulawa

Tired of blasting invaders from outer space? This whimsical game is set in a very different world—the miniature jungle in your own backyard. The original version was written for the Atari. We've added versions for Apple, TI, Commodore 64, VIC-20, and IBM PC and PCjr computers. A joystick is required for the Atari and Commodore 64 versions.

Guide Webster, the hungry tree spider, in his endless search for a square meal. Roving back and forth across his tree limb, he watches for bugs to appear in the grass below. When the time is right, he drops down on a strand of silk for a light snack, then climbs back up his web to look for more.

Unfortunately, this backyard paradise isn't quite perfect. The more Webster eats, the faster the bugs move, making it harder to find the next meal. Even worse, he's not the only one with an appetite—there's a speedy scorpion sharing the same hunting ground, stealing bugs when he can and giving Webster a sting whenever he drops too close.

Atari Version

Program 1—the Atari version of "Webster Dines Out"—will run on any Atari computer with at least 32K memory. Use the joystick to move Webster left or right at the top of the screen. When a bug passes below, press the button to make him drop down.

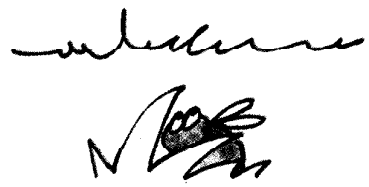
Your goal is to score points as quickly as possible. Each bug is worth 25 points and you get 50 bonus points for snaring two bugs in a single drop. Webster has three lives in each game; getting stung by the scorpion costs you a life but does not reduce your score. The scorpion is a tough competitor: When Webster drops down, the scorpion speeds up to increase his chances of stealing a bug.

There are six skill levels, each harder than the last. As you advance to higher levels, the bugs and scorpion speed up, the grass grows longer, and a grey rock appears in the lawn. The other creatures hide behind these objects, but Webster can drop behind them too. The game ends when you lose all three lives or exhaust your time at the highest skill level.

Commodore 64 And VIC-20 Versions

Both Commodore versions of Webster Dines Out are scored like the Atari game—25 points for each bug, with a 50 point bonus for capturing two at once. You begin with three lives, and lose one each time you collide with the scorpion.

The 64 version (Program 2) is played with a joystick in port 2. The bugs and scorpion move across a sloping, multicolored lawn; at higher skill levels, colorful objects grow up to obscure your view of the ground. Play ends when your lives



are used up or time runs out at the last skill level.

The VIC-20 version of Webster Dines Out is written in machine language (ML) for the unexpanded VIC. Program 3 is a BASIC loader that saves the ML program on disk or tape. Since the loader won't fit in an unexpanded VIC, you'll need at least 8K memory expansion to run it (a Commodore 64 can also be used; see instructions below). Type in and save Program 3, but don't try to run it yet. Enter this line in direct mode (without a line number):

```
POKE6609,0:POKE43,209:POKE44,25:
NEW
```

Now reload Program 3 and run it. Press D to save the game on disk, or T to save it on tape. The finished program will be named WEBSTER (replacing any other program of that name on your disk). If you don't have memory expansion, you can use a Commodore 64 to create the VIC game (of course, the game itself runs only on a VIC). To run the loader on a 64, change the 57809 to 57812, and 63109 to 62957 in lines 9–11 of Program 3. Then follow the procedure described above.

Once the game is saved, remove any memory expansion and load and run it like a BASIC program. Move Webster with the < and > keys, and drop him down by pressing the space bar. As you progress to higher skill levels, the speed increases, and rocks appear below, blocking your vision. You can snare bugs from behind the rocks, but be careful not to drop onto a hidden scorpion. Play continues until you lose all three lives.

IBM Version

Program 4 runs on any PCjr with cartridge BASIC and any IBM PC with BASICA and a color/graphics adapter card. Press the left and right cursor keys to move Webster, and the space bar to drop.

Webster's lifeline is displayed at the top of the screen. When you drop to get a bug, your energy level is drained and your lifeline shrinks. Capturing a bug restores your energy and expands your lifeline. You'll score 10 points for catching a beetle, and 20 for each bug, with bonus points for multiple captures. Extra bonus points are awarded at the 1,000, 5,000 and 10,000 point marks. As your score increases, the bugs speed up and become more scarce; your energy will drain faster, too. The game ends when you hit the scorpion or your energy drains to zero.

Apple Version

This version of Webster Dines Out will run on any Apple II series computer. Since it's written entirely in machine language (ML), it must be entered using the "Apple MLX" machine language editor found elsewhere in this issue. MLX will greatly simplify the usually tedious job of accurately entering the many numbers that make up a ML program. But be sure that you read the MLX article and understand how to use MLX before you begin entering the data from Program 5.

When you run MLX, it will ask for a starting and ending address. Use the values indicated in Program 5:

```
START ADDRESS?    1100
END ADDRESS?      1F14
```

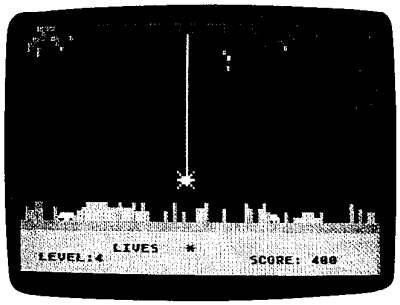
MLX will then give you a menu of options. Choose E for enter and give 1100 as the starting address. A prompt for the first line will appear, and you can begin entering the data from Program 5. If you don't type the entire listing in one sitting, follow the instructions in the MLX article for saving a partially complete version and reloading it later. When you're finished typing, MLX will prompt you for a filename for the completed machine language program. To load and run the game, simply type BRUN"WEBSTER" (or whatever name you used for the completed program) and press RETURN.

The scoring is identical to the Atari version. Use the left and right arrow keys to move on the branch, and press the space bar to drop to the ground. Avoid colliding with the giant grasshopper—when that happens, Webster loses a life (and is carried bodily off the screen). The grass in the lawn grows higher as the game progresses, making your job more difficult. You can drop into the grass to snare a hidden bug, but be sure to keep track of the giant hopper, who might be lurking there as well.

TI Version

This version of Webster (Program 6) uses sprites for the spider, bugs, and scorpion, and thus requires TI Extended BASIC. You can use either joystick or keyboard controls. For the keyboard, press the S key to move left, the D key to move right, and the space bar to make Webster drop. Scoring is identical to the Atari version: You have three lives in each game, and six possible levels. Bugs are worth 25 points, with a 50 point bonus for catching two at once.

At the bottom of the screen you'll see Webster's lifeline. Dropping for a bug drains your energy and shrinks the lifeline; catching a bug restores your energy to normal. You lose a life whenever your energy drains to zero or you hit the scorpion. The game will not end until you lose all three lives.



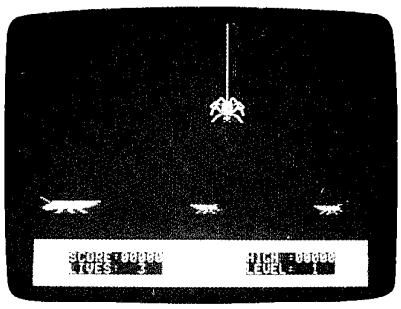
Growing grass and grey rocks make it tough for Webster to find a meal in Atari "Webster Dines Out."

Program 1: Webster Dines Out For Atari

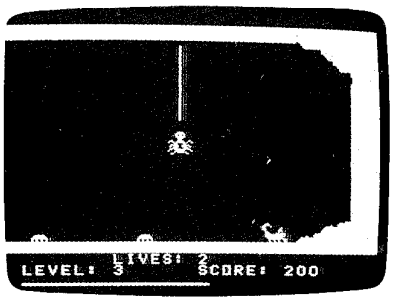
Please refer to "COMPUTE!'s Guide to Typing In Programs" before entering this listing.

```
KI 10  GOSUB 7000
KK 20  GOSUB 7100
LA 30  GOSUB 7600
OO 50  LEVEL=LEVEL+1
CL 80  ON LEVEL GOSUB 85,1000
       0,10100,10200,10300,10
       400,19000
PL 85  POKE 656,2:POKE 657,8:
       ? LEVEL
LH 90  POKE 53248,XPOS:PD0$(Y
       POS,YPOS+LEN(SPIDER$))
       =SPIDER$
PO 98  COUNTER=0
DP 99  REM *** MAIN LOOP STAR
       T ***
CA 100 S=STICK(0)
NC 110 XPOS=XPOS+4*(S=7)-4*(
       S=11)
OC 144 IF XPOS>XMAX THEN XPO
       S=XMAX
NO 146 IF XPOS<XMIN THEN XPO
       S=XMIN
```

```
0F A0
07 B8
03 A4
00 C1
00 51
03 DA
7F 87
7F 3D
00 0F
3B 0D
3A 28
08 F4
7F E1
00 14
00 9F
63 45
00 DA
6E 6B
6E F2
22 DD
7F CE
00 F2
00 43
08 49
40 E8
00 8A
38 BA
40 05
15 99
1E 49
0E 91
3A C2
60 98
60 25
06 61
0C 84
78 B9
7F 3E
18 B2
00 B2
40 2D
00 5A
1F D2
40 EE
0E 14
7C E2
03 05
1F 4D
20 D7
20 3C
20 44
20 4C
20 54
20 5C
20 64
20 6C
20 74
20 7C
20 84
20 8C
20 94
20 9C
20 A4
20 AC
0F B4
0F BC
0F C4
0F CC
0F D4
0F DC
0F E4
0F EC
0F F4
0F FC
0F 05
0F 0D
0F 15
0F 1D
0F 25
0F 2D
00 F3
00 E8
7F F0
00 B2
```



Avoid the mammoth grasshopper in Apple "Webster Dines Out."



"Webster Dines Out" for TI uses sprites for the spider, scorpion and bugs.

Program 6: Webster Dines Out For TI

Translation by Patrick Parrish, Programming Supervisor

```
100 RANDOMIZE :: CALL CLEAR
    :: GOSUB 600 :: CALL S
    CREEN(2):: CALL MAGNIFY
    (3)
110 CALL HCHAR(1,1,114,32):
    : FOR T=30 TO 32 :: CAL
    L VCHAR(1,T,114,20):: N
    EXT T :: CALL VCHAR(3,2
    9,114,2)
120 CALL HCHAR(19,29,114,2)
    :: CALL HCHAR(20,28,114
    ,3):: CALL HCHAR(3,28,1
    17):: CALL HCHAR(4,29,1
    17)
130 CALL HCHAR(2,29,114)::
    CALL HCHAR(2,27,114,2)
140 CALL HCHAR(2.1.115.25):
    : CALL HCHAR(2,26,117)
150 CALL HCHAR(19,28,112)::
     CALL HCHAR(19,29,113):
    : CALL HCHAR(20,26,112)
    :: CALL HCHAR(20,27,113
    )
160 CALL VCHAR(5,30,118)::
    CALL VCHAR(6,30,116,13)
    :: CALL HCHAR(21,1,104,
    32):: OX=117 :: OY=9
170 LEVEL=1 :: LIVES=3 :: S
    CORE=0 :: E=0 :: BUGFL(
    3)=0 :: BUGFL(4)=0
180 CALL SPRITE(#2,100,5,14
    5,1+RND*256,0,-15*RND+1
    0)
190 DISPLAY AT(22,9):"LIVES
    :";LIVES :: DISPLAY AT(
    23,2):"LEVEL:";LEVEL ::
     DISPLAY AT(23,16):"SCO
    RE:";SCORE
200 YPOS=OY :: XPOS=OX :: C
    ALL VCHAR(YPOS/8+2,XPOS
    /8+1,32,16)
210 CALL SPRITE(#1,136,14,Y
    POS,XPOS):: CCOL=25 ::
    CALL HCHAR(24,4,120,25)
220 FOR L=3 TO 4 :: IF BUGF
    L(L)=0 AND RND<.5 THEN
    CALL SPRITE(#L,96,-(L=3
    )*11-(L=4)*5,145,1+198*
    RND,0,-5+RND*10):: BUGF
    L(L)=1
230 NEXT L :: CALL MOTION(#
    2,0,-15*RND+10*LEVEL/2)
240 CALL KEY(0,K,S):: IF S=
    0 THEN CALL JOYST(1,XR,
    YR):: XR=SGN(XR)ELSE XR
    =(K=83)-(K=68)
250 COUNT=COUNT+LEVEL :: IF
     COUNT<10 THEN 330
260 CALL HCHAR(24,4+CCOL,32
    ):: COUNT=0 :: CCOL=CCO
    L-1 :: IF CCOL<>3 THEN
    280
270 FOR K=1 TO 30 STEP 2 ::
     CALL SOUND(50,1175,K):
    : NEXT K
280 IF CCOL<>-1 THEN 330
290 LIVES=LIVES-1 :: CALL S
    PRITE(#1,140,14,YPOS,XP
    OS):: FOR I=3 TO 17
300 CALL LOCATE(#1,YPOS+I*8
    ,XPOS):: CALL SOUND(25,
    (23-I)*20,3):: IF I<>17
     THEN 320
310 FOR L=1 TO 800 :: NEXT
    L
320 NEXT I :: IF LIVES=0 TH
    EN 710 ELSE 190
330 XPOS=XPOS-SGN(XPOS-5)*X
    R*(XR=-1)*16-SGN(197-XP
    OS)*XR*(XR=1)*16
340 CALL LOCATE(#1,YPOS,XPO
    S)
350 FOR L=3 TO 4 :: CALL MO
    TION(#L,0,-5+RND*10)::
    NEXT L
360 CALL KEY(1,K,S):: CALL
    KEY(0,K1,S):: IF (K<>18
    )*(K1<>32)THEN 220 ELSE
     COUNT=10
370 FOR I=3 TO 18 :: CALL H
    CHAR((YPOS+(I-1)*8)/8,X
    POS/8+1,128):: CALL LOC
    ATE(#1,YPOS+I*8,XPOS)
380 IF I=18 THEN CALL COINC
    (ALL,C)ELSE 510
390 IF C=0 THEN 510
400 FOR L=2 TO 4 :: CALL MO
    TION(#L,0,0):: NEXT L
410 FL=0 :: FOR L=2 TO 4 ::
     CALL COINC(#1,#L,16,C)
420 IF C=0 THEN 470
430 IF L=2 THEN 450
440 FOR K=9 TO 21 STEP 3 ::
     CALL SOUND(10,-1,K)::
    NEXT K :: CALL DELSPRIT
    E(#L):: E=2 :: FL=FL+1
    :: BUGFL(L)=0 :: GOTO 4
    70
450 CALL SPRITE(#1,140,14,1
    44,XPOS):: LIVES=LIVES-
    1 :: E=1 :: CALL SOUND(
    50,-6,4):: FOR K=5 TO 1
    5 STEP 5 :: CALL SOUND(
    10,-5,K):: NEXT K
460 FOR T=1 TO 600 :: NEXT
    T :: L=4
470 NEXT L :: IF E=0 OR E=1
     THEN 510
480 PTS=FL*25 :: PTS=-(PTS=
    25)*25-(PTS=50)*100 ::
    SCORE=SCORE+PTS :: DISP
    LAY AT(3,22):SCORE
490 IF SCORE>=100*LEVEL THE
    N LEVEL=LEVEL+1+(LEVEL=
    6):: DISPLAY AT(23,8):L
    EVEL;
500 CCOL=25 :: CALL HCHAR(2
    4,4,120,25):: E=0
510 NEXT I
520 IF E=1 THEN CALL DELSPR
    ITE(#1):: CALL VCHAR(YP
    OS/8+2,XPOS/8+1,32,16):
    : IF LIVES=0 THEN 710 E
    LSE E=0 :: GOTO 190
530 FOR I=17 TO 2 STEP -1 :
    : CALL LOCATE(#1,YPOS+I
    *8,XPOS):: CALL HCHAR((
    YPOS+I*8)/8,XPOS/8+1,32
    ):: NEXT I :: CALL LOCA
    TE(#1,YPOS,XPOS)
540 CALL COINC(ALL,C):: IF
    C=0 THEN 220
550 FOR I=2 TO 4 :: CALL MO
    TION(#I,0,0):: NEXT I
560 FOR L=3 TO 4 :: CALL CO
    INC(#2,#L,10,C1)
570 IF C1=0 THEN 590
580 FOR I=5 TO 15 STEP 5 ::
     CALL SOUND(10,-5,I)::
    NEXT I :: CALL DELSPRIT
    E(#L):: BUGFL(L)=0
590 NEXT L :: GOTO 220
600 FOR T=112 TO 118 :: REA
    D A$ :: CALL CHAR(T,A$)
    :: NEXT T
610 CALL CHAR(96,"000000000
    0000000000000070F1F1212
    0000000000000000000000E
    0F0F84848"):: REM BUG
620 CALL CHAR(100,"00003078
    C4C0E06030383B3F1F0F090
    4000000000001038204EFA
    FEF8F52291"):: REM SCOR
    PION
630 CALL CHAR(104,"FFFFFFFF
    FFFFFFFF",128,"10101010
    10101010",120,"00000000
    0000FFFF")
640 CALL CHAR(136,"03070507
    030163970C3E4E1C6F870B3
    0C0E0A0E0C080C6E9307C72
    38F6E1D00C"):: REM WEBS
    TER
650 CALL CHAR(140,"00000000
    204448482F1F3F7B787B3F1
    F0000000010884848D0E0F7
    7D7F77E0C0"):: REM WEBS
    TER2
660 DATA 000000031F3F7FFF,0
    00F1F1FFFFFFFFF,FFFFFFF
    FFFFFFFFF,FFFFFFFF000000
    000,0F0F0F0F0F0F0F0F0F
670 DATA FFFFFF3F1F0F0F07,0
    F0F0F0707070707
680 FOR I=2 TO 8 :: CALL CO
    LOR(I,15,1):: NEXT I
690 CALL COLOR(1,15,2,10,3,
    1,11,7,1,12,5,1,13,16,1
    )
700 RETURN
710 DISPLAY AT(22,15):LIVES
     :: DISPLAY AT(10,9):"G
    AME OVER";:: DISPLAY AT
    (12,5):"PLAY AGAIN (Y/N
    )?";
720 CALL KEY(0,K,S):: IF S=
    0 THEN 720
730 IF K=78 OR K=110 THEN E
    ND
740 IF K=89 OR K=121 THEN C
    ALL HCHAR(10,9,32,11)::
     CALL HCHAR(12,7,32,19)
    :: GOTO 170 ELSE 720
```

REVIEWS

The Hitchhiker's Guide To The Galaxy

Neil Randall

Requirements: Commodore 64 with a disk drive; Atari 400/800, XL, or XE with at least 48K RAM and a drive; Apple II-series computer with at least 48K RAM and a drive; Apple Macintosh; IBM PC, PCjr, or MS-DOS 2.0 compatible computer with at least 48K RAM and a drive; TI-99/4A with 48K RAM, a disk drive, and Extended BASIC or Mini-Memory or Editor/Assembler cartridge; or a Kaypro II with CP/M. Versions for the Apricot and Epson QX-10 are forthcoming.

The Hitchhiker's Guide to the Galaxy may well be Infocom's best effort to date. There are several reasons for this. First, the comic absurdity of Douglas Adams' popular radio/television/novel series translates well to Infocom's style of interactive fiction. Second, the story has a built-in sense of humor, which increases the player's enjoyment and reduces frustration. Third, the story itself is fascinating.

A best-selling novel and hit BBC radio series, later adapted for TV, *The Hitchhiker's Guide to the Galaxy* follows the hilarious adventures in space of Arthur Dent. Dent is an ordinary Englishman who one day witnesses the destruction of Earth (to make room for an Intergalactic Bypass). He eventually gets caught in the problem of finding the ultimate question to Life, the Universe, and Everything. The story is filled with absurd characters and wonderfully illogical events.

The narrative nature of *The Hitchhiker's Guide* is well suited to Infocom's text-only adventure format. In addition, Infocom's software boasts the industry's most advanced parser, that part of an adventure program which interprets the commands you enter. This means you can enter commands as normal English sentences and generally the computer will understand you. Infocom adventures take a long time to play, mainly because each contains several major puzzles you must figure out.

Comic Relief

Infocom's version of the story begins like the original series. Playing the role of Arthur Dent, you awaken to the sight of a bulldozer about to demolish your home. Solving this puzzle is quite easy, but the next major puzzle, aboard the Vogon spacecraft, is more difficult. In order to understand all alien languages, you have to find a way to get a Babel Fish into your ear (honest!). Although somewhat frustrating, this puzzle is entirely true to the humor of the radio series, and even if you don't solve it, you'll get several good laughs.

Humor, in fact, is the game's saving grace. It distinguishes *The Hitchhiker's Guide* from several other Infocom adventures. Most Infocom games take a long, long time to play, and for the most part you are simply solving puzzles. After awhile the puzzles may become frustrating, and in desperation you may begin seeking out other people to assist you in your struggles.

Not so with *The Hitchhiker's Guide*. I am committed to solving the thing myself, since I believe I have as small a grasp on logic as did the original series. I am far enough into the adventure to report that the game's humor consistently prevents you from becoming too frustrated. Adams' humor is sprinkled throughout, in descriptions (one object you find is "the thing which your aunt gave you which you don't know what it is") and in the actions of the characters and robots (Marvin the Paranoid Android never fails to elicit a laugh). For Infocom's version of *The Hitchhiker's Guide* to be successful, it had to be consistently funny and consistently absurd. Happily, it is both.

It also had to diverge from the series in one major respect: Arthur Dent's role had to change from spectator to major participant. In the original story, Dent is swept along by the strange happenings around him. But interactive fiction is strongest when your character can, to some degree, affect those happenings. The role of passive observer does not translate well to an adventure program.

If it ever does, *The Hitchhiker's Guide to the Galaxy* may be redone with a different emphasis. But until then, Infocom has given us a thoroughly enjoyable rendition of a delightfully bizarre story. Recommended for all adventure gamers.

The Hitchhiker's Guide to the Galaxy
Infocom, Inc.
55 Wheeler Street
Cambridge, MA 02138
$34.95 (Atari & Commodore 64)
$39.95 (all other versions)

Super-Text

Arthur Leyenberger

Requirements: Commodore 64 with a disk drive; Atari 400/800, XL, or XE with at least 48K RAM and a drive; Apple II-series computer with at least 48K RAM and a drive (80-column card optional); IBM PC with at least 48K, a drive, and DOS 1.1 (not compatible with the PCjr). A printer is highly recommended. The version reviewed was for the Atari; other versions are similar.

According to recent surveys, word processing is second only to entertainment as the primary application for most home computers. Whether you're jotting a short letter to Aunt Viola or compiling a term paper, word processing can make your writing less painful and even enjoyable.

There are scores of word processors available for computers these days. Your chief criterion for selecting one should be that it has the functions you require to accomplish your writing tasks. It is also important to consider your future needs so you won't outgrow your word processor.

THE BEGINNER'S PAGE

Tom R. Halfhill, Editor

FOR-NEXT Applications

Last month we covered the basics of looping with the FOR-NEXT statement. Now let's take a look at some practical applications of this essential technique.

FOR-NEXT is such a general-purpose structure, it has numerous uses. Here's an example of how you might apply it in part of a checkbook-balancing program that sums the amounts for a month's worth of checks:

```
10 PRINT "HOW MANY CHECKS
   THIS MONTH";
20 INPUT CH
30 FOR X=1 TO CH
40 PRINT "AMOUNT OF CHECK";
50 INPUT AM
60 SUM=SUM+AM
70 NEXT X
80 PRINT "TOTAL AMOUNT IS
   $";SUM
```

Let's take a careful look at this program. Try running it. Line 10 prompts the user to enter the number of checks to be added; line 20 stores the response in the variable CH. Line 30 is a little tricky. It marks the beginning of the loop with a FOR statement as shown in last month's examples, but the number of repetitions specified is a variable, not a number.

The variable, CH, contains the number of checks the user entered in response to the prompt. Therefore, the number of times the FOR-NEXT loop will repeat depends on the user's response. In effect, the program adapts itself to the user's needs. Think of what would happen if you specified the number of loops with a certain number, say 10. If the user has only 7 checks, the program would make too many loops, demanding amounts for 3 checks that were never written. If the user has 23 checks, the program would add only 10 of them together, ignoring the remaining 13.

Line 40 prompts the user to enter the dollar amount of the first check (do not type a dollar sign). Line 50 stores the response in the variable AM. Line 60 creates the variable SUM to keep track of the total and adds AM to SUM (when the first pass through the loop begins, SUM equals 0).

Line 70 marks the end of the loop. It circles back to the FOR statement on line 30. Now the loop begins its second pass. Again, a prompt asks the user to input a check amount. Again, the response is stored in the variable AM (replacing the previous amount for the first check). Again, AM is added to SUM, so SUM now contains the cumulative amounts of the first and second checks. And again, at line 70, NEXT X circles back to line 30 for the third pass.

This continues until the number of loops specified by CH is reached. Then the loop is done and the program proceeds to line 80. The program prints the total amount held in SUM and ends.

This example shows two things: first, the usefulness of FOR-NEXT loops when combined with other techniques, such as INPUT statements; and second, the flexibility of FOR-NEXT loops when modified slightly, such as specifying the number of loops with a variable instead of a particular number.

Speed READing

One of the most common applications of FOR-NEXT loops is to combine them with the READ and DATA statements. (If you aren't familiar with READ-DATA, don't worry; it's a subject for a future column.) By embedding a READ statement within a loop, you can efficiently fill an array with DATA, or POKE numbers for custom character sets or machine language subroutines into memory.

Let's try a simple example. Say you're writing some sort of calendar program that requires the computer to print a column of numbers representing the number of days in each month of the year. Without looping, you could take this approach:

```
10 PRINT "31"
20 PRINT "28"
30 PRINT "31"
40 PRINT "30"
50 PRINT "31"
60 PRINT "30"
70 PRINT "31"
80 PRINT "31"
90 PRINT "30"
100 PRINT "31"
110 PRINT "30"
120 PRINT "31"
```

But a FOR-NEXT loop with READ-DATA is much more efficient:

```
10 FOR X=1 TO 12
20 READ A
30 PRINT A
40 NEXT X
50 DATA 31,28,31,30,31,30,
   31,31,30,31,30,31
```

Line 10 sets up a FOR-NEXT loop with 12 passes (the number of elements in the DATA statement at line 50). Line 20 reads a number from the DATA statement and stores it in the variable A.

Line 30 prints the value of A, which changes after each pass through the loop. During the first pass, A equals 31 (the number of days in January) because 31 is the first DATA element. During the second pass, A equals 28 (the number of days in February) because 28 is the second DATA element. This continues for all 12 passes, concluding when A is assigned the value 31 for the twelfth month, December.

Next month, we'll continue our discussion of FOR-NEXT by showing how to put loops within loops, and even why you might want to create a loop that does *absolutely nothing*. We'll also cover some variations of FOR-NEXT in different versions of BASIC. ©

PROGRAMMING THE TI

C. Regena

Multiple Choice Test

I've seen a number of computer programs written for multiple choice tests. The computer is an ideal way to administer such tests because it can mix up the test questions so each run is different. However, all the programs I have seen always print the choices in the same order. This month's program is a general-purpose multiple choice tester that randomly arranges both the questions (without repetition) *and* the possible answers.

This program can be used for questions on any topic. Computer literacy questions are included here for an example.

The questions and answers are in DATA statements. Each DATA statement contains six items. The first item is the question; the next four are the possible answers; and the last item is the number of the correct answer. The final DATA statement signals the end of the question list:

```
1350 DATA ZZZ,Z,Z,Z,Z,0
```

You may use any number of possible questions that will fit in the computer's memory. Line 190 is a DIMension statement that allows for 30 possible questions. To increase the number of questions, change all the 30s in line 190.

Reading The Data

Line 160 defines the number of questions in the test, and line 300 performs the procedure for the specified number of questions. If you want to alter this number, change the 20 in these two lines plus the printed score in line 690. Also make sure you have as many or more questions and answers in the DATA statements as you want in the test.

The variable I is used as a counter for the questions. Questions read from the DATA statements are stored in the string variable T$, the four possible answers are stored in A$, and the number of the correct answer is stored in B. These values are in arrays to keep the answers with the corresponding questions.

As the information is being read in, S$(I) is set equal to A for use as a signal so questions won't be repeated during the quiz. When a question X is printed, S$(X) is set equal to "" (null). Line 320 chooses a random number X, but if S$(X) is null, the question has previously been used and a different X must be chosen. Line 350 prints the question.

Lines 370–390 define C(J) for the four answers to mix up the order in which the answers are printed. Line 400 randomly chooses D for the correct answer. The C variable for the correct answer is set to zero so it cannot be used in another position. Lines 430–490 mix up the order of the answers, making sure the correct answer is in the right position and each answer is used only once. Lines 500–530 print the four answers with the possible choices A, B, C, and D.

Lines 540–580 receive the student's answer, making sure it is a letter from A to D, and then print the choice. Line 590 checks to see if the key pressed is the correct choice. Line 600 prints the message for an incorrect answer, then prints the correct answer. Line 620 prints CORRECT for a correct answer, and line 630 increments the score, SC. Lines 640–670 wait for the student to press ENTER before going to the next question.

Lines 680–700 clear the screen, then print the score.

To customize the test, simply change the questions and answers in the DATA statements, making sure you have enough questions for a complete quiz and that the last DATA statement contains ZZZ to signal the end. You might also prefer a fancier title screen.

Here is an example of changing the DATA statements. Suppose your question is "In which year did Columbus discover America?" with the possible answers 1256, 1492, 1776, and 1812. The correct answer is in the second position. The DATA statement would look like this:

```
720 DATA IN WHICH YEAR DID
COLUMBUS DISCOVER AMERICA?
730 DATA 1256,1492,1776,1812,2
```

If you want to save typing effort, you can obtain a copy of this program by sending a cassette or disk, a stamped, self-addressed mailer, and $3 to:

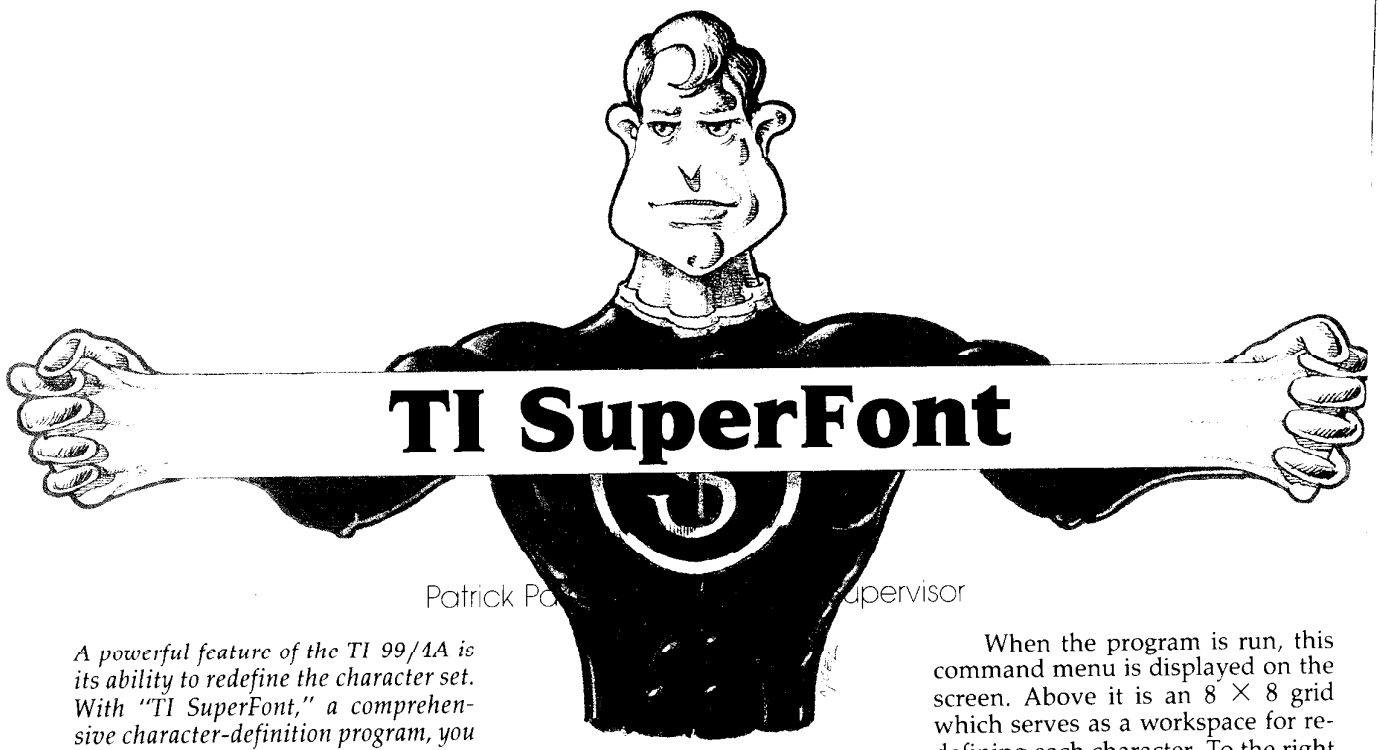
C. Regena
P.O. Box 1502
Cedar City, UT 84720

Please be sure to specify the name of the program and that you need the TI version.

Multiple Choice Test

```
100 REM  MULTIPLE CHOICE TE
    ST(5 SPACES)
110 CALL CLEAR
120 PRINT "**********************
    **********"
130 PRINT "*  MULTIPLE CHOI
    CE TEST  *"
140 PRINT "**********************
    **********"
150 PRINT ::::::
160 PRINT "TEST OF 20 QUEST
    IONS"
170 PRINT ::"PRESS LETTER O
    F CORRECT"
180 PRINT :"ANSWER FOR EACH
     QUESTION."
190 DIM T$(30),A$(30,4),B(3
    0),S$(30),AA$(4)
200 I=1
210 READ T$(I),A$(I,1),A$(I
    ,2),A$(I,3),A$(I,4),B(I
    )
220 IF T$(I)="ZZZ" THEN 260
230 S$(I)="A"
240 I=I+1
250 GOTO 210
260 I=I-1
270 PRINT ::"PRESS <ENTER>
    TO START."
```

```
280 CALL KEY(0,K,S)
290 IF K<>13 THEN 280
300 FOR P=1 TO 20
310 RANDOMIZE
320 X=INT(I*RND)+1
330 IF S$(X)="" THEN 320
340 CALL CLEAR
350 PRINT T$(X)::
360 S$(X)=""
370 FOR J=1 TO 4
380 C(J)=1
390 NEXT J
400 D=INT(4*RND)+1
410 AA$(D)=A$(X,B(X))
420 C(B(X))=0
430 FOR J=1 TO 4
440 IF J=D THEN 490
450 E=INT(4*RND)+1
460 IF C(E)=0 THEN 450
470 AA$(J)=A$(X,E)
480 C(E)=0
490 NEXT J
500 FOR J=1 TO 4
510 PRINT CHR$(64+J);". ";A
    A$(J)
520 NEXT J
530 PRINT ::
540 CALL SOUND(100,1497,2)
550 CALL KEY(0,K,S)
560 IF (K<65)+(K>68)THEN 55
    0
570 CALL HCHAR(23,3,K)
580 PRINT
590 IF K=64+D THEN 620
600 PRINT "NO, THE ANSWER I
    S ";CHR$(64+D);"."
610 GOTO 640
620 PRINT "CORRECT"
630 SC=SC+1
640 PRINT :"PRESS <ENTER>."
650 CALL KEY(0,K,S)
660 IF K<>13 THEN 650
670 NEXT P
680 CALL CLEAR
690 PRINT "OUT OF 20 QUESTI
    ONS,"
700 PRINT "YOUR SCORE IS ";
    SC::::
710 GOTO 1360
720 DATA ONE OF THE MAJOR A
    TTRACTIONSOF A COMPUTER
     IS THAT IT
730 DATA HAS ACTIVE INVOLVE
    MENT.,IS EXPENSIVE.,IS
    A STATUS SYMBOL.
740 DATA ALLOWS UNINVOLVEME
    NT.,1
750 DATA A VIDEO GAME IS BE
    ST(8 SPACES)DESCRIBED A
    S A(N)
760 DATA EXPENSIVE TOY.,SPE
    CIAL PURPOSE COMPUTER.,
    HOME COMPUTER.,EDUCATIO
    NAL TOY.,2
770 DATA THE COMPUTER OWES
    ITS(7 SPACES)FLEXIBILIT
    Y TO THE FACT THATIT IS
780 DATA SMALL.,COMPLICATED
    .,PROGRAMMABLE.,AN ELEC
    TRONIC DEVICE.,3
790 DATA "BECAUSE A COMPUTE
    R IS(7 SPACES)PROGRAMMA
    BLE,"
800 DATA IT CAN BE USED TO
    PERFORM(3 SPACES)ONLY A
     LIMITED NUMBER OF
    (4 SPACES)FUNCTIONS.
810 DATA IT CANNOT BE USED
    FOR(7 SPACES)EDUCATIONA
    L PURPOSES.
820 DATA IT CANNOT BE USED
    FOR(7 SPACES)ENTERTAINM
    ENT.
830 DATA IT CAN BECOME A GE
    NERAL(5 SPACES)PURPOSE
    TOOL.,4
840 DATA THE MAIN ADVANTAGE
     OF A(5 SPACES)COMPUTER
     AS OPPOSED TO
    (6 SPACES)OTHER CALCULA
    TING DEVICES(3 SPACES)I
    S ITS
850 DATA COST.,SIZE.,PORTAB
    ILITY.,PROGRAMMABLE NAT
    URE.,4
860 DATA BOOKS AND MANUALS
    THAT(6 SPACES)ACCOMPANY
     A COMPUTER-RELATEDPROD
    UCT ARE
870 DATA SOFTWARE.,DOCUMENT
    ATION.,DATA.,COMPU-FORM
    S.,2
880 DATA VISICALC IS BEST D
    ESCRIBED  AS A(N)
890 DATA TUTORIAL PROGRAM.,
    ELECTRONIC SPREADSHEET.
900 DATA EDUCATIONAL PROGRA
    M.,ENTERTAINMENT PROGRA
    M.,2
910 DATA ALL OF THE FOLLOWI
    NG ARE(4 SPACES)PROGRAM
    MING LANGUAGES EXCEPT
920 DATA BASIC.,PASCAL.,VIS
    ICALC.,LOGO.,3
930 DATA ONE OF THE MAJOR P
    ROBLEMS INACQUIRING COM
    PUTER LITERACY IS
940 DATA PEOPLE NEED TO BE
    SKILLED(3 SPACES)IN MAT
    HEMATICS TO USE
    (7 SPACES)COMPUTERS.
950 DATA THE COMPUTER IS A
    VERY(6 SPACES)COMPLICAT
    ED MACHINE.
960 DATA THE FIELD HAS ITS
    OWN(7 SPACES)LEXICON OR
     LANGUAGE.
970 DATA PEOPLE NEED A BACK
    GROUND(4 SPACES)IN LOGI
    C AND STATISTICS.,3
980 DATA THE PARTS OF A COM
    PUTER ARE ARRANGED IN S
    UCH A WAY AS TOFORM A(N
    )
990 DATA SYSTEM.,MACHINE.,S
    UBSYSTEM.,ORGANIZATION.
    ,1
1000 DATA THE PROCESSING OF
     DATA IN A COMPUTER SY
    STEM RESULTS IN  THE G
    ENERATION OF
1010 DATA A PROGRAM.,READOU
    TS.,INFORMATION.,STATI
    STICS.,3
1020 DATA "BASICALLY, A COM
    PUTER IS(4 SPACES)INTE
    NDED TO PRODUCE"
1030 DATA INFORMATION.,DATA
    .,STATISTICS.,PROGRAMS
    .,1
1040 DATA THE BASIC FUNCTIO
    N OF A(5 SPACES)COMPUT
    ER IS TO TRANSFORM
1050 DATA PROGRAMS INTO DAT
    A.
1060 DATA DATA INTO PROGRAM
    S.
1070 DATA INFORMATION INTO
    DATA.
1080 DATA DATA INTO INFORMA
    TION.,4
1090 DATA "BY USING A -----
    -, ONE MAY  CONNECT A
    COMPUTER TO THE
    (3 SPACES)TELEPHONE TO
     PERMIT COMPUTERCONFER
    ENCING."
1100 DATA ADAPTER,CONNECTOR
    ,CONFERENCE LINK,MODEM
    ,4
1110 DATA INTANGIBILITY IS
    A MAJOR(4 SPACES)CHARA
    CTERISTIC OF
1120 DATA SOFTWARE.,THE COM
    PUTER.,HARDWARE.,MAGNE
    TIC DISKS.,1
1130 DATA THE USE TO WHICH
    A COMPUTER IS PUT IS C
    ALLED A(N)
1140 DATA PROGRAM.,ROUTINE.
    ,APPLICATION.,FUNCTION
    .,3
1150 DATA INSIDE THE COMPUT
    ER(9 SPACES)INFORMATIO
    N IS REPRESENTED  BY
1160 DATA PUNCHED CARDS.,EL
    ECTRONIC SIGNALS.,MAGN
    ETIC TAPE.,MAGNETIC DI
    SKS.,2
1170 DATA THE ON/OFF PATTER
    N THAT IS  USED IN THE
     COMPUTER IS THE BASIS
     OF THE
1180 DATA CIRCUIT CODE.,BIN
    ARY CODE.,BINOMIAL COD
    E.,DIDECIMAL CODE.,2
1190 DATA "WITH TELECOMMUTI
    NG,(9 SPACES)INFORMATI
    ON IS MOST COMMONLYTRA
    NSMITTED BETWEEN
    (9 SPACES)TERMINALS"
1200 DATA BY RADIO.,OVER TE
    LEPHONE WIRES.
1210 DATA VIA SATELLITE.,BY
     TELEVISION.,2
1220 DATA A COMPUTER PROGRA
    M IS AN(4 SPACES)EXAMP
    LE OF
1230 DATA HARDWARE.,SOFTWAR
    E.,FIRMWARE.,FLEXWARE.
    ,2
1240 DATA THE FIRST ELECTRO
    NIC(8 SPACES)COMPUTER
    WAS
1250 DATA ENIAC.,ENID.,IBM
    MARK I.,IBM CYBERNAUGH
    T.,1
1260 DATA THE COMPUTER IS I
    NSTRUCTED  OR TOLD WHA
    T TO DO BY
1270 DATA HARDWARE.,FIRMWAR
    E.,SOFTWARE.,SMARTWARE
    .,3
1280 DATA THE MOST SIGNIFIC
    ANT FACTOR IN PURCHASI
    NG A COMPUTER IS
1290 DATA RELATIVE COST.,AV
    AILABLE SOFTWARE.,AVAI
    LABLE HARDWARE.,AVAILA
    BLE FIRMWARE.,2
1300 DATA WHICH IS THE MOST
    COMMON(4 SPACES)TYPE
    OF SECONDARY STORAGE
    (3 SPACES)CURRENTLY US
    ED IN PERSONAL  COMPUT
    ERS?
1310 DATA FLOPPY DISKS,BUBB
    LE MEMORY,ELECTRIC CON
    DUCTORS,TUNNEL JUNCTIO
    N MEMORY,1
1320 DATA RAM IS USED AS A
    MEASURE OF
1330 DATA PRIMARY STORAGE C
    APACITY.,PROCESSING PO
    WER.
1340 DATA PROCESSING SPEED.
    ,WORD LENGTH.,1
1350 DATA ZZZ,Z,Z,Z,Z,0
1360 END
```



TI SuperFont

Patrick Pa[illegible]upervisor

A powerful feature of the TI 99/4A is its ability to redefine the character set. With "TI SuperFont," a comprehensive character-definition program, you can harness this capability. Requires Extended BASIC and a joystick (printer optional).

The character graphics capabilities of the TI-99/4A are well known. But to redefine a character on the TI by the usual means (see the *TI User's Reference Guide*, pages II-76 to II-79), you must follow a tedious, multistep procedure. First, you plot the character in an 8 × 8 grid. Next, you convert each row of the grid into a two-digit hexadecimal number and then sequentially combine the numbers from each row to generate a *pattern identifier*, or coded representation of the character. Finally, you place this pattern identifier along with a chosen ASCII value for the character in a CALL CHAR statement. Anyone who has repeatedly endured this process can attest to its drudgery.

Fortunately, the process is easily computerized, and several character-definition programs have been written for the TI. Until now, however, these programs have not taken full advantage of the TI's capabilities. With "TI SuperFont" (Program 1), once-tedious character manipulations can now be undertaken with ease.

Sixteen Commands

SuperFont, originally written for the Atari by Charles Brannon, first appeared in the January 1982 issue of COMPUTE! and featured 18 commands for redefining characters. After using this outstanding program on several occasions, I was convinced that TI users deserved a utility as versatile and convenient. That's how TI SuperFont was born.

In converting SuperFont, a few commands with less value to the TI user were eliminated while certain more practical commands were added. As it turned out, the real challenge was to fit the program into a TI without memory expansion. The final version leaves only a hundred or so bytes to spare. However, certain adjustments must be made if you are using a disk drive with the program. Before loading and running SuperFont, type CALL FILES(1). This will free up memory ordinarily reserved for additional disk file manipulation.

TI SuperFont offers the following 16 commands or modes:

E	EDIT	N	INPUT
R	RESTORE CH	H	RESTORE CHSET
F	COPY	W	WRITE DATA
M	MIRROR	V	REVERSE
A	ROTATE	C	CLEAR
I	INSERT	D	DELETE
L	LOAD FONT	S	SAVE FONT
P	PRINT CH	T	PRINT CHSET

When the program is run, this command menu is displayed on the screen. Above it is an 8 × 8 grid which serves as a workspace for redefining each character. To the right of the grid, the current mode and, in some cases, a prompt will be displayed. Below this is printed the entire TI character set (codes 32–143) with each subset (eight characters) denoted by a different background color. (If you find the colors annoying, remove the FOR–NEXT loop in line 300.)

Several commands require that you pick a character from the TI character set. In these instances, a box-shaped sprite (CHR$(143)) appears over the last character referenced from the set (defaults to space). Position the sprite with the joystick over the desired character and press the fire button. Unless indicated otherwise, each command returns you to the *EDIT* mode upon completion.

Now let's examine each command, beginning with *EDIT*. (The ALPHA-LOCK key should be up when making menu selections.)

- *EDIT* is the basic editing command. After you press E, SuperFont requests you to choose a character from the character set. The character selected is copied into the grid and the box-shaped sprite appears. This is actually like a cursor, controlled with the joystick. Press the fire button to set a point (if a point is clear) or reset a point (if a point is already set). You can draw lines by holding down the button while moving the joystick. When you're pleased with

the appearance of the shape in the grid, press ENTER to redefine the character. (To completely redesign a character from scratch, use the *CLEAR* command, described below.)

- *INPUT* lets you type in a pattern identifier and assign it to a particular character code. After selecting *INPUT*, choose a character from the set with the joystick and then type in the hexadecimal code for the redefined character. The hexadecimal code can be typed in upper- or lowercase (a routine at line 960 automatically converts the code to uppercase). The *INPUT* command is handy when attempting to associate a pattern identifier with a CHR$ code in someone else's program.
- *RESTORE CH* restores the current character to its original configuration. This command is useful if you've mangled a character or changed the wrong one.
- *RESTORE CHSET* restores the entire character set to its initial appearance.
- *COPY* copies a character to a second location in the character set. SuperFont prompts you for the first character (the one to be moved) and the second character (the destination character). This command is handy for arranging your customized characters to fit the various color codes.
- *WRITE DATA* displays the pattern identifier for each selected character along with its ASCII value. Very handy when comparing characters or for providing a few character codes for another program.
- *MIRROR* produces a mirror image of the current character in the grid.
- *REVERSE* puts the current character in the grid in reverse field: All dots become blanks, and all blanks become dots.
- *ROTATE* turns the current character 90 degrees clockwise.
- *CLEAR* completely clears out the current character. For creating new characters from scratch.
- *INSERT* places a row of blanks in the current character. Move the cursor in the grid with the joystick to the row where you wish to insert the blanks and press ENTER. All rows below will scroll down and the bottom row will be lost.
- *DELETE* is the opposite of *INSERT*. Position the cursor on a row in the grid and press ENTER. The row will be eliminated and all other rows will scroll upward. *DELETE* and *INSERT* can be used with *ROTATE* to scroll characters left or right in the grid (of course, one row will be lost in both cases).
- *LOAD FONT* loads a previously SAVEd character set (a font) from tape or disk. SuperFont prompts you for the device and filename. Be sure to type this in the standard format (that is, CS1 or DSK1.FILENAME). Again, capital letters need not be used. The routine that converts from lower- to uppercase takes care of this for you. If you're using tape, the screen will be restored after the tape system messages have been printed (the same occurs with *SAVE FONT*, discussed below). When loading is complete, a command prompt appears.
- *SAVE FONT* saves to tape or disk (in a data file format) only those characters which have been altered since SuperFont was run. Since each character code is saved as a separate record, a large set may take 30 minutes to save. As with *LOAD FONT*, you will be prompted for the device and filename. If you accidentally hit L (for *LOAD FONT*) or S from the main menu, simply press ENTER to abort the command when prompted for the device and filename.

Once saved, character sets can be loaded into any program where they're needed (we'll consider this in greater detail shortly). As with *LOAD FONT*, a command prompt appears when the operation is complete.

- *PRINT CH* prints the current character in an 8 × 8 grid along with its ASCII and pattern identifier codes, then returns you to the main menu. Be sure to modify line 1260 to correspond to the specifications of your printer.
- *PRINT CHSET* is the same as the previous command, except it prints every character which has been modified.

The commands offered by SuperFont are versatile, but you may want to add others. Since the program uses most of the TI's memory, unless you have additional RAM you'll have to substitute your own routine for an existing one. Fortunately, the program is modular in structure. Just follow the branching IF statements from line 360 to 920 for the current commands. If you do alter the program, test it thoroughly to make sure you still have plenty of memory left.

Redesigning a character with "TI SuperFont."

Retrieving A Font Or Screen

After you've saved a newly created character set, how do you go about recovering it for use in another program? Program 2 shows how this is done.

In line 130, the device and filename for the character set file is defined as B$ (the filename used here is FONT). If you store this file on tape rather than disk, line 130 should read B$="CS1". Lines 140–160 load in the new character set and print it on the screen. Line 170 sets up a delay so you can see that the character set has successfully loaded.

With SuperFont, you can perform many chores with ease. You can customize your character set (ever wished for true lowercase?), create graphics characters and animated figures (space creatures!), or just play around. The uses of this utility are endless. I'm sure you'll have as much fun discovering them as I have.

Program 1: TI SuperFont

```
10 DIM A$(111),C$(15),N$(11
   2),D(15),V(8,8):: L=32
20 E=15 :: Q$="DEVICE.FILEN
   AME?" :: GOSUB 1240 :: G
   OTO 260
30 F=0 :: GOSUB 40 :: GOTO
   340
40 CALL HCHAR(5,14,L,16)::
   RETURN
50 CALL HCHAR(3,17,L,7):: C
   ALL HCHAR(7,17,L,16):: R
   ETURN
60 FOR I=5 TO 7 :: CALL HCH
```

```
   AR(I,13,L,18):: NEXT I :
   : RETURN
70 Z$=N$(W-L)
80 FOR I=0 TO 15 :: D(I)=AS
   C(SEG$(Z$,I+1,1))-48 ::
   D(I)=D(I)+(D(I)>9)*7
90 NEXT I :: J=0 :: FOR I=0
    TO 7 :: DISPLAY AT(2+I,
   1):C$(D(J));:: DISPLAY A
   T(2+I,5):C$(D(J+1));:: J
   =J+2 :: NEXT I :: RETURN
100 CALL DELSPRITE(#1):: DI
    SPLAY AT(5,15):"WAIT"
110 FOR R=1 TO 8 :: FOR C=1
    TO 8
120 IF M=109 THEN CALL GCHA
   R(R+1,11-C,H):: GOTO 15
   0
130 IF M=97 THEN CALL GCHAR
   (10-C,R+2,H):: GOTO 150
140 CALL GCHAR(R+1,2+C,H)
150 V(R,C)=H-141 :: NEXT C
   :: NEXT R
160 H$="0123456789ABCDEF" :
   : IF M=118 THEN H$="FED
   CBA9876543210"
170 Z$="" :: FOR R=1 TO 8 :
   : LO=V(R,5)*8+V(R,6)*4+
   V(R,7)*2+V(R,8)+1
180 HI=V(R,1)*8+V(R,2)*4+V(
   R,3)*2+V(R,4)+1
190 Z$=Z$&SEG$(H$,HI,1)&SEG
   $(H$,LO,1):: NEXT R
200 IF (M<>100)*(M<>105)THE
   N 240
210 IF M<>100 THEN 230
220 Z$=SEG$(Z$,1,ROW*2-2)&S
   EG$(Z$,ROW*2+1,14)&"00"
    :: GOTO 240
230 Z$=SEG$(Z$,1,ROW*2-2)&"
   00"&SEG$(Z$,ROW*2-1,16-
   ROW*2)
240 CALL CHAR(W,Z$):: N$(W-
   L)=Z$ :: IF (M=100)+(M=
   105)THEN GOSUB 70
250 GOSUB 40 :: RETURN
260 F$="0000000010010001101000
   0010101100111100010011010
   1010111100110111101111"
270 FOR I=0 TO 15 :: Z$=SEG
   $(F$,I*4+1,4):: D$=""
280 FOR J=1 TO 4 :: T=VAL(S
   EG$(Z$,J,1))+141 :: D$=
   D$&CHR$(T):: NEXT J ::
   C$(I)=D$ :: NEXT I
290 CALL CHAR(141,"",142,RP
   T$("F",16),143,"FFA1A1A
   1818181FF"):: FOR I=141
    TO 143 :: CALL CHARPAT
   (I,A$(I-L)):: N$(I-L)=A
   $(I-L):: NEXT I
300 CALL DELSPRITE(#1):: CA
   LL CLEAR :: FOR I=2 TO
   14 :: CALL COLOR(I,2,I+
   2):: NEXT I
310 FOR I=L TO 143 :: PRINT
    CHR$(I);:: NEXT I :: D
   ISPLAY AT(1,11):"SUPERF
   ONT" :: GOSUB 1100
320 FOR R=1 TO 8 :: CALL HC
   HAR(R+1,3,141,8):: NEXT
    R
330 BR=20 :: BC=2 :: W=L
340 CALL SOUND(100,800,2):
    DISPLAY AT(3,15):"MODE
   ?"
350 CALL KEY(0,M,S):: IF S=
   0 THEN 350
360 IF M<>101 THEN 510
370 T=1 :: GOSUB 1200 :: GO
   SUB 980 :: IF (F=1)*(KK
   >112)THEN 30 ELSE IF K=
   112 THEN M=K :: GOSUB 4
   0 :: GOTO 910
380 GOSUB 70 :: Z=1
390 CALL SPRITE(#1,143,10,9
   ,17):: R=1 :: C=2 :: CA
   LL GCHAR(R+1,C+1,T)
400 CALL KEY(0,K,S):: IF (K
   =13)+(K=112)THEN ROW=R
   :: GOSUB 100 :: GOSUB 1
   180 :: IF K<>112 THEN O
   N Z GOTO 340,590
410 IF (K>13)THEN M=K :: GO
   TO 360
420 CALL JOYST(1,X,Y):: IF
   ABS(X)+ABS(Y)=8 THEN 42
   0
430 CALL KEY(1,KK,S):: IF (
   KK<>18)*(ABS(X)+ABS(Y)=
   0)THEN 400
440 OK=0 :: IF ABS(X)+ABS(Y
   )=4 THEN OK=1
450 C=C-(X=4)+(X=-4):: R=R-
   (Y=-4)+(Y=4)
460 C=C-(C=1)*8+(C=10)*8 ::
    R=R-(R=0)*8+(R=9)*8
470 CALL LOCATE(#1,8*R+1,8*
   C+1)
480 IF (KK=18)*(OK=0)THEN C
   ALL GCHAR(R+1,C+1,T)::
   T=283-T
490 IF (OK=1)*(KK<>18)THEN
   CALL GCHAR(R+1,C+1,T)
500 CALL HCHAR(R+1,C+1,T)::
    CALL SOUND(-1,294,3)::
    GOTO 400
510 IF M<>110 THEN 570
520 T=1 :: GOSUB 1200 :: GO
   SUB 980 :: IF F=1 THEN
   30
530 DISPLAY AT(5,12):"CHAR
   HEX CODE?" :: ACCEPT AT
   (6,11)SIZE(16)BEEP:D$ :
   : IF LEN(D$)<>16 THEN 5
   30
540 GOSUB 60 :: GOSUB 960
550 N$(W-L)=Z$ :: GOSUB 80
   :: CALL CHAR(W,Z$)
560 GOSUB 40 :: GOTO 590
570 IF M<>114 THEN 600
580 GOSUB 1200 :: CALL CHAR
   (W,A$(W-L)):: N$(W-L)=A
   $(W-L)
590 Z=1 :: GOSUB 40 :: GOSU
   B 70 :: M=101 :: GOSUB
   1180 :: CALL HCHAR(3,17
   ,69):: CALL SOUND(50,88
   0,3):: GOTO 390
600 IF M<>104 THEN 620
610 GOSUB 1200 :: FOR I=L T
   O 143 :: CALL CHAR(I,A$
   (I-L)):: N$(I-L)=A$(I-L
   ):: NEXT I :: GOTO 590
620 IF M<>102 THEN 670
630 GOSUB 1200
640 DISPLAY AT(5,15):"1ST C
   HAR?" :: GOSUB 980 :: I
   F F=1 THEN 30 ELSE TM=W
650 GOSUB 70 :: DISPLAY AT(
   5,15):"2ND CHAR?" :: GO
   SUB 980 :: IF F=1 THEN
   30 ELSE CALL DELSPRITE(
   #1)
660 CALL CHARPAT(TM,Z$):: C
   ALL CHAR(W,Z$):: N$(W-L
   )=Z$ :: GOTO 590
670 IF M=109 THEN GOSUB 120
   0 :: GOSUB 100 :: GOTO
   590
680 IF M=118 THEN GOSUB 120
   0 :: GOSUB 100 :: GOTO
   590
690 IF M<>97 THEN 730
700 GOSUB 1200
710 GOSUB 100 :: GOSUB 70 :
   : GOSUB 1180 :: T=0 ::
   D$="AGAIN (Y/N)?" :: GO
   SUB 1220 :: GOSUB 40 ::
    IF T=1 THEN 710
720 GOTO 590
730 IF M=99 THEN GOSUB 1200
    :: D$=RPT$("0",16):: C
   ALL CHAR(W,D$):: N$(W-L
   )=D$ :: GOTO 590
740 IF M=105 THEN GOSUB 120
   0 :: Z=2 :: GOTO 390
750 IF M=100 THEN GOSUB 120
   0 :: Z=2 :: GOTO 390
760 IF M<>119 THEN 820
770 T=1 :: GOSUB 1200 :: GO
   SUB 980 :: IF F=1 THEN
   F=0 :: GOTO 810 ELSE GO
   SUB 70
780 DISPLAY AT(7,16):"CHAR=
   ";W :: DISPLAY AT(9,11)
   :N$(W-L)
790 D$="AGAIN(Y/N) ?" :: GO
   SUB 1220
800 CALL HCHAR(9,11,L,18)::
    IF T=1 THEN GOSUB 40 :
   : GOTO 770
810 GOSUB 60 :: GOTO 340
820 IF M<>108 THEN 860
830 GOSUB 1200
840 GOSUB 940 :: OPEN #1:D$
   ,INTERNAL,INPUT ,FIXED
850 INPUT #1:T,N$(T):: IF T
   <>112 THEN CALL CHAR(T+
   L,N$(T)):: GOTO 850 ELS
   E CLOSE #1 :: GOSUB 60
   :: IF ASC(D$)=67 THEN 3
   00 ELSE 340
860 IF M<>115 THEN 910
870 GOSUB 1200 :: GOSUB 940
880 OPEN #1:D$,INTERNAL,OUT
   PUT,FIXED :: FOR I=L TO
    143
890 IF N$(I-L)<>A$(I-L)THEN
    PRINT #1:I-L,N$(I-L)
900 NEXT I :: T=112 :: F$="
   " :: PRINT #1:T,F$ :: C
   LOSE #1 :: GOSUB 60 ::
   IF ASC(D$)=67 THEN 300
   ELSE 340
910 IF M=112 THEN H=1 :: GO
   SUB 1260
920 IF M=116 THEN H=0 :: GO
   SUB 1260
930 GOTO 340
940 DISPLAY AT(5,13):Q$ ::
   ACCEPT AT(6,14):D$ :: I
   F D$="" THEN GOSUB 60 :
   : GOTO 340 ELSE GOSUB 9
   60
950 RETURN
960 Z$="" :: FOR I=1 TO LEN
   (D$):: F$=SEG$(D$,I,1):
   : IF (ASC(F$)>96)*(ASC(
   F$)<123)THEN F$=CHR$(AS
   C(F$)-L)
970 Z$=Z$&F$ :: NEXT I :: D
   $=Z$ :: RETURN
980 CALL SPRITE(#1,143,10,B
   R*8+1,BC*8+1)
990 CALL JOYST(1,X,Y):: IF
   ABS(X)+ABS(Y)=8 THEN 99
   0
1000 BC=BC-(X=4)+(X=-4):: W
   =W-(X=4)+(X=-4)
1010 BR=BR-(Y=-4)+(Y=4):: W
   =W-(Y=-4)*28+(Y=4)*28
1020 IF BC<2 THEN BC=29 ::
   BR=BR-1
1030 IF BC>29 THEN BC=2 ::
   BR=BR+1
1040 IF BR<20 THEN BR=23 ::
    W=W+112
1050 IF BR>23 THEN BR=20 ::
    W=W-112
1060 CALL KEY(1,KK,ST):: CA
   LL KEY(0,K,S)
```

```
1070 IF S<>0 THEN F=1 :: IF
     M=111 THEN RETURN ELS
     E CALL DELSPRITE(#1)::
     RETURN
1080 IF KK=18 THEN CALL SOU
     ND(100,110,2):: GOSUB 4
     0 :: CALL DELSPRITE(#1
     ):: RETURN
1090 GOTO 980
1100 DISPLAY AT(11,1):"E ED
     IT";TAB(14);"N INPUT"
1110 DISPLAY AT(12,1):"R RE
     STORE CH";TAB(14);"H R
     ESTORE CHSET"
1120 DISPLAY AT(13,1):"F CO
     PY";TAB(14);"W WRITE D
     ATA"
1130 DISPLAY AT(14,1):"M MI
     RROR";TAB(14);"V REVER
     SE"
1140 DISPLAY AT(15,1):"A RO
     TATE";TAB(14);"C CLEAR
     "
1150 DISPLAY AT(16,1):"I IN
     SERT";TAB(14);"D DELET
     E"
1160 DISPLAY AT(17,1):"L LO
     AD FONT";TAB(14);"S SA
     VE FONT"
1170 DISPLAY AT(18,1):"P PR
     INT CH";TAB(14);"T PRI
     NT CHSET" :: RETURN
1180 FOR I=0 TO 5 STEP 2 ::
      CALL HCHAR(7,17+I,W):
     : NEXT I :: RETURN
1190 R=20 :: C=2 :: W=L ::
     CALL SPRITE(#1,143,2,R
     *8+1,C*8+1):: RETURN
1200 GOSUB 50 :: CALL HCHAR
     (3,17,M-L):: IF T=1 TH
     EN DISPLAY AT(5,15):"P
     ICK A CHAR" :: T=0
1210 RETURN
1220 DISPLAY AT(5,15):D$ ::
      ACCEPT AT(5,27)BEEP V
     ALIDATE("yn")SIZE(1):Z
     $ :: IF Z$="y" THEN T=
     1
1230 RETURN
1240 CALL CLEAR :: CALL SCR
     EEN(E):: DISPLAY AT(12
     ,7):"LOADING CHARPATS"
      :: FOR I=127 TO 140 :
     : CALL CHAR(I,""):: NE
     XT I
1250 FOR I=L TO 140 :: CALL
      CHARPAT(I,A$(I-L)):: 
     N$(I-L)=A$(I-L):: NEXT
      I :: RETURN
1260 DISPLAY AT(3,15):"PRIN
     T" :: OPEN #1:"RS232/2
     .BA=9600.DA=8.PA=N"
1270 TM=W :: IF H=1 THEN 13
     00
1280 FOR T=L TO 143 :: IF N
     $(T-L)<>A$(T-L)THEN W=
     T ELSE 1350
1290 E=E+1 :: E=(E=17)*14+E
     :: CALL SCREEN(E)
1300 IF ((F=1)*(H=1))+(H=0)
     THEN GOSUB 70 :: GOSUB
      1180
1310 FOR R=2 TO 9 :: IF R=5
      THEN PRINT #1:TAB(5);
     "CHR$ # - "&"<"&STR$(T
     )&">";
1320 PRINT #1:TAB(30);:: FO
     R C=3 TO 10 :: CALL GC
     HAR(R,C,X):: IF X=141
     THEN X=45 ELSE X=88
1330 PRINT #1:CHR$(X);:: NE
     XT C :: IF R=5 THEN PR
     INT #1:TAB(47);"HEX CO
     DE - "&"<"&N$(W-L)&">"
1340 NEXT R :: PRINT #1 ::
     PRINT #1 :: IF H=1 THE
     N 1360
1350 NEXT T
1360 CLOSE #1 :: F=0 :: H=0
      :: E=15 :: W=TM :: CA
     LL SCREEN(E):: RETURN
```

Program 2: Character Set Loader

```
100  !GAME
110  !GET REDEFINED CHARS
120  CALL CLEAR
130  B$="DSK1.FONT"
140  OPEN #1:B$,INTERNAL,INP
     UT ,FIXED
150  INPUT #1:F,NEWA$ :: IF
     F<>112 THEN CALL CHAR(F
     +32,NEWA$):: PRINT CHR$
     (F+32);:: GOTO 150
160  CLOSE #1
170  FOR T=1 TO 1000 :: NEXT
      T
```