Forcing Function:

The forcing function is an expression implemented using an *Expression Object*. The following variables have been added to the functionality of the *Expression Object* by Inclination:

```
time   - The current time
t      - The current time
mass   - Mass of the respective block
m      - Mass of the respective block
x      - The current x position of the respective block
y      - The current y position of the respective block
vx     - The x velocity of the respective block
vy     - The y velocity of the respective block
ax     - The x acceleration of the respective block
ay     - The y acceleration of the respective block
nx     - The x normal force of the respective block
```

```
ny      - The y normal force of the respective block
ex      - The current x external force of the respective block
ey      - The current y external force of the respective block
fx  - The current x frictional force of the respective block
fy  - The current y frictional force of the respective block
angle   - The current angle of the respective wedge
```

Expression Concepts:

The expression object is a versatile parser.   The order of operations is standard.

Any expression can be surrounded by parenthesis. The categories described below should demonstrate the correct syntax for the parser.

Built-in-operators:

Operator	Description
+	binary addition
-	binary subtraction, unary negation
/	binary division
*	binary multiplication

```
%                   binary modulus
^                   binary exponential
```

Conditionals:

Conditionals evaluate an expression for TRUE or FALSE (1 or 0). If the expression is TRUE the expression to the left of the ELSE is returned, otherwise the conditional to the right of the ELSE is returned.

Conditional Format

(expression) IF expression THEN expression ELSE expression ENDIF

Conditional Operators	Description
`<`	`less than`
`>`	`greater than`
`<=`	`less than or equal`
`>=`	`greater than or equal`
`==`	`equal`
`!=`	`not equal`
`!`	`(unary) negation`
`||`	`logical or`
`&&`	`logical and`

Associativity and Order of Operations:

Operators are listed in order of precendence with operators of equal precedence on the same line.

Operator	Description
^	right to left
- !	right to left

```
*  /  %                    left ro right
+   -                      left to right
<   >  <=  >=              left to right
==   !=                    left to right
&&                         left to right
||                         left to right
```

Constants:

Constant	Value
E	2.7182818284590452354

```
LOG2E           1.4426950408889634074
LOG10E          0.43429448190325182765
LN2             0.69314718055994530942
LN10            2.30258509299404568402
PI              3.14159265358979323846
PI_2            1.57079632679489661923
PI_4            0.78539816339744830962
1_PI            0.31830988618379067154
2_PI            0.63661977236758134308
2_SQRTPI                1.12837916709551257390
SQRT2           1.41421356237309504880
SQRT1_2         0.70710678118654752440
MAXDOUBLE       1.7976931348623157e308
```

```
MINDOUBLE          4.9406564584124654e-324
```

Numbers:

Real Numbers (treated as double floating point numbers internally)

```
1.2
1.
.0
1.2e-12
```

```
1e-13
-1e-11
-1.2e-11
+1.2e+11
1.2e15
```

Integer Numbers (treated as integers internally)

```
1
2
22
-89
```

Built-in-functions:

Functions requiring a parameter list follow the format:

```
function(param1, param2, ..., paramn) .
```

Functions requiring 0 parameters may be typed as:

```
function()
```
or

```
function .
```

Most functions use floating point numbers. The functions will attempt to use the correct representation of the numbers regardless of the actual C parameter type.

Function	No of Parameters	C-Function Called
abs	1	abs (integer)
acos	1	acos
acosh	1	acosh
asin	1	asin

asinh	1	asinh
atan	1	atan
atanh	1	atanh
atan2	2	atan2
betai	3	Internally implemented
cbrt	1	cbrt
ceil	1	ceil
copysign	2	copysign
cos	1	cos
cosh	1	cosh
drem	2	drem
erf	1	erf
erfc	1	erfc

```
exp        1        exp
expm1      1        expm1
fabs       1        fabs (floating point)
finite     1        finite
floor      1        floor
fmod       2        fmod
gammap     2        Internally implemented
gammaq     2        Internally implemented
hypot      2        hypot
jn         2        jn
j0         1        j0
j1         1        j1
lgamma     1        lgamma
```

```
ln          1          log
logb        1          logb
log1p       1          log1p
log10       1          log10
pow         2          pow
rand        0          random (is called then divided
                            by MAXINT)

rint        1          rint
scalb       2          scalb
sin         1          sin
sinh        1          sinh
sqrt        1          sqrt
tan         1          tan
```

```
tanh            1           tanh
yn              2           yn
y0              1           y0
y1              1           y1
```