```
TO RUN: make tom
                //**Note sometimes the makefile does not execute
properly, just keep running the same line until it does
KEYWORDS IN OUR LANGUAGE (NOT CASE SENSITIVE)
        MAXX [integer]
                //Use this to override the maxium x range, as a part
of graph
                //Default starts at 10

        MAXY [integer]
                //Use this to override the maxium y range, as a part
of graph
                //Default starts at 10

        MINX [integer]
                //Use this to override the minimum x range, as a part
of graph
                //Default starts at 0

        MINY [integer]
                //Use this to override the minum y range, as a part
of graph
                //Default starts at 0

        GRAPH [expression]
                //The graph uses the maxx, maxy, minx, miny to grath
an expression
                //An expression is a function to f(x)
                //Example expressions are: x^2 or x^5 + 9*x + 2 or 5

        SOLVELIN [ax + b = cx + d] or [ax + b = cx] or [ax + b = d]
                //Prints the numerical solution to a linear equation
in the forms above
                //Example equations are: 5x+2=0 or 10x + 2 = 4x + 3
or 1x+0=1
                //**Note that all parts must be added, and have
numerical values, 1 and 0 are valid options

        SOLVEQUAD [ax^2 + bx + c = d]
                //Prints the numerical solution(s) to a a quadratic
equation in the form above
                //Example equations are: 1x^2+0x+-4=0 or 10/3x^x + 2x
+ 1 = 9
                //**Note that all parts must be added, and have
numerical values, 1 and 0 are valid options

        SOLVECUBE [ax^3 + bx^2 + cx + d = e]
                //Prints the numerical solution(s) to a a cubic
equation in the form above
                //Example equations are: 1x^2+0x+-4x+=0 or 10/3x^x +
```

```
2x + 1 = 9
                //**Note that all parts must be added, and have
numerical values, 1 and 0 are valid options

        WHILE [condition { newline statement(s) }]
                //Repeats statemet(s) while a condition is true
                //A condition is in the following form: [expression
boolean_operator expression]
                //An expression can also be an integer value or a
variable
                //Boolean operators are ! != == < > <= >=
                //Parentheses do not surround the conditions
                //**Note that the first bracket must be on the same
line as the conditions
                //Statements can contaion a single statement or
multiple

        FOR [variable = expression, condition, variable = expression
{ newline statement(s) }]
                //Repeats statement(s) a variable amount of times
                //A condition is in the following form: [expression
boolean_operator expression]
                //**Note that the variable must already be defined
                //An expression can also be an integer value or a
variable
                //Boolean operators are != == < > <= >=
                //Parentheses do not surround the conditions
                //**Note that the first bracket must be on the same
line as the conditions
                //Statements can contaion a single statement or
multiple

        IF [condition { newline statement(s) }]
                //If a condition is true it will run a set of
statement(s)
                //A condition is in the following form: [expression
boolean_operator expression]
                //An expression can also be an integer value or a
variable
                //Boolean operators are != == < > <= >=
                //Parentheses do not surround the conditions
                //**Note that the first bracket must be on the same
line as the conditions
                //Statements can contaion a single statement or
multiple
                //**Note a newline is required after the last
statement in a list between the brackets
                //**Note a newline is required after the closing
bracket
```

```
        ELSE IF [condition { newline statement(s) }]
                //Alternate path if another condition is true it will
run a set of statement(s)
                //A condition is in the following form: [expression
boolean_operator expression]
                 //An expression can also be an integer value or a
variable
                //Boolean operators are != == < > <= >=
                //Parentheses do not surround the conditions
                //**Note that the first bracket must be on the same
line as the conditions
                //Statements can contaion a single statement or
multiple
                 //**Note a newline is required after the closing
bracket

        ELSE [{ newline statement(s) }]
                 //Runs a set of statement(s) if all of the above
conditions are false
                //**Note that the first bracket must be on the same
line as the conditions
                //Statements can contaion a single statement or
multiple
                 //**Note a newline is required after the closing
bracket

        INT [variable_name]
                 //Defines a variable of type integer
                 //Variable names contain only letters A-Z,a-z
                 //**Note that variables cannot be initalized on the
same line

        FLOAT [variable_name]
                 //Defines a variable of type floating point
                 //Variable names contain only letters A-Z,a-z
                 //**Note that variables cannot be initalized on the
same line

        PUT [variable] or [string]
                 //Prints the variable value to the screen
                 //Or prints a string to the screen
                 //**Note strings must be in quotations

        GET [variable]
                 //Gets a integer or floating point value from the
user
                 //**Note variable must be defined

        ALPHA (REQUIRED)
                 //Indicates the staring point to the program
```

```
        OMEGA (REQUIRED)
                //Indicates the ending point to the program
```

EXAMPLE PROGRAM:

```
alpha
 minx -5
 maxx 5
 miny -5
 maxy 5

 put "Example 1\n"

 int a

 a = 0

 graph x^3

 while a<10{
         get a
         if a>0 {
                graph 2*x^3 + -4 * x^2 + -22 *x +24
        }
        solvecube 2x^3 + -4x^2 + -22x + 24 = 0
        solvecube 3x^3 + -10x^2 + 14x + 27 = 0
}

 graph x^2 + 2*x + -1
 solvequad 1x^2 + 2x + 0 = 1
 put "Done\n"

omega
```