Version 3 of the QUAD1 Program

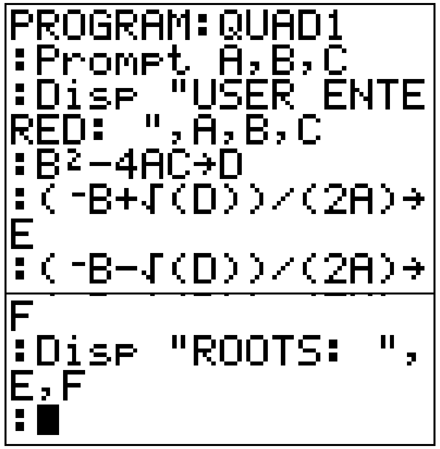
Objective: Edit the program named QUAD1
add error checking to make the program
more robust.

This will demonstrate the following key components of program development:

1) That you are able to create a program
2) That you are able to accept user input
3) That you are able to send the user some output
4) That you are able to run (or execute) a program that you created.
5) That you are able to perform some calculations data processing) on input provided from the user.
6) Handling invalid as well as invalid data
7) Processing groups of statements "conditionally" as well as unconditionally.

These are valuable first steps for developing programs in any programming language, on any computing platform.

Version 3 of the QUAD1 Program



```
PROGRAM:QUAD1
:Prompt A,B,C
:Disp "USER ENTE
RED: ",A,B,C
:B²-4AC→D
:(⁻B+√(D))/(2A)→
E
:(⁻B-√(D))/(2A)→
F
:Disp "ROOTS: ",
E,F
:█
```

Versions 1 and 2 of the QUAD program execute each line of the program in order from the first line Prompt to the last line Disp. After executing each line the program ends.

Try to execute the QUAD1 program using invalid data:
For example, the calculations for E and F divide by 2A,
if A is equal to 0 then you should get a divide by 0 error.

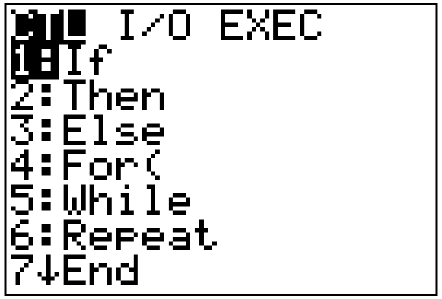
If B^2 - 4AC is less than 0, then you will be trying to take the square root of a negative number - which might be fine if you are dealing with complex numbers and you are looking for imaginary numbers, but of you are looking for real solutions then B^2 - 4AC needs to not be negative.

Version 3 of the QUAD1 Program

In version 3 of the QUAD1 program we will add some conditional process and some error checking. First, the user enters 0 for A, send them an error message and do not try to run the rest of the program. An outline of the program would look like this (is pseudo-code):

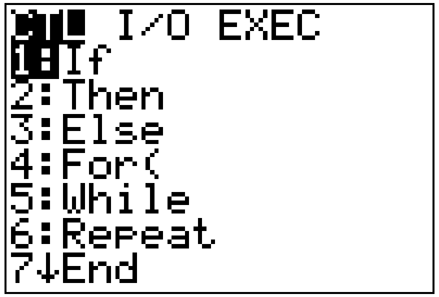
```
Prompt the user for A, B, C
If A = 0
   then
         tell the user that 0 for A is not valid
         stop the program
  else
       the value for A is not 0,
       calculate D, E, and F
       tell the user that the roots are E and F.
```

The CTL prgm menu supplies the commands that are needed to conditionally execute block of statements.



```
CTL I/O EXEC
1:If
2:Then
3:Else
4:For(
5:While
6:Repeat
7↓End
```

Version 3 of the QUAD1 Program



In this version of the program we will group the statements of the program into blocks - some will be executed unconditionally and some will be executed conditionally.

```
Prompt the user for A, B, C
If A = 0
   then
         tell the user that 0 for A is not valid
         stop the program
  else
       the value for A is not 0,
       calculate D, E, and F
       tell the user that the roots are E and F.
```

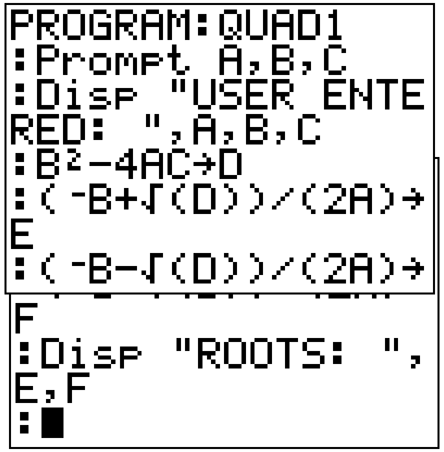
Executed unconditionally

Executed conditionally
... only when $A = 0$

Executed conditionally
... only when A is NOT 0

Now test the program, both using values for $A = 0$ and $A \neq 0$.

Version 3 of the QUAD1 Program



```
PROGRAM:QUAD1
:Prompt A,B,C
:Disp "USER ENTE
RED: ",A,B,C
:B²-4AC→D
:(-B+√(D))/(2A)→
E
:(-B-√(D))/(2A)→
F
:Disp "ROOTS: ",
E,F
:█
```

Step 1: Add the lines

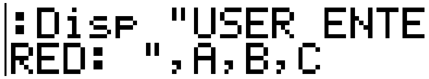
If A = 0

Then

before the line

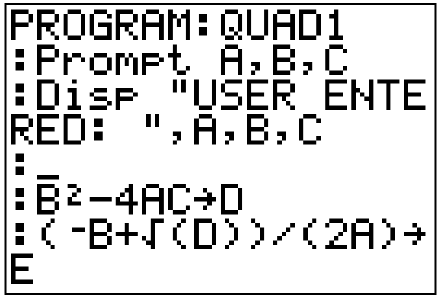
```
:B²-4AC→D
```

Move the cursor to the end of the line



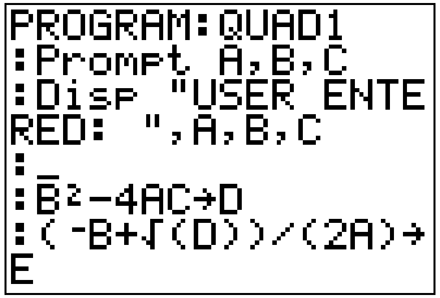
```
:Disp "USER ENTE
RED: ",A,B,C
```

To put the calculator in insert mode hit INS which is the 2nd function on the [del] key: 2nd DEL ENTER



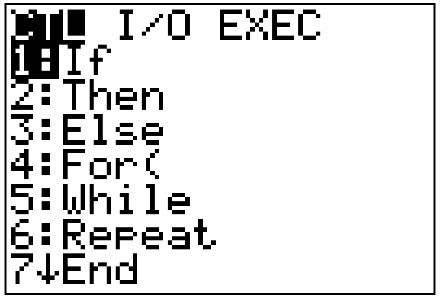
```
PROGRAM:QUAD1
:Prompt A,B,C
:Disp "USER ENTE
RED: ",A,B,C
:_
:B²-4AC→D
:(-B+√(D))/(2A)→
E
```

Version 3 of the QUAD1 Program



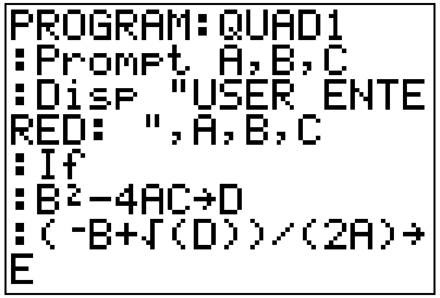
```
PROGRAM:QUAD1
:Prompt A,B,C
:Disp "USER ENTE
RED: ",A,B,C
:_
:B²-4AC→D
:(-B+√(D))/(2A)→
E
```

PRGM ENTER



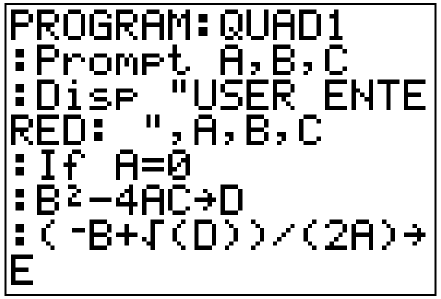
```
CTL I/O EXEC
1:If
2:Then
3:Else
4:For(
5:While
6:Repeat
7↓End
```

To add the If statement go to the prgm menu, and select If from the CTL menu. It is the top item and will already be highlighted.



```
PROGRAM:QUAD1
:Prompt A,B,C
:Disp "USER ENTE
RED: ",A,B,C
:If
:B²-4AC→D
:(-B+√(D))/(2A)→
E
```

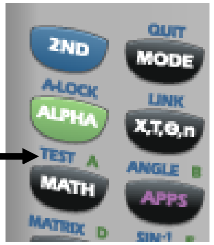
To add: A = 0 you will need the = test operator which you can find in the test menu above the MATH key:

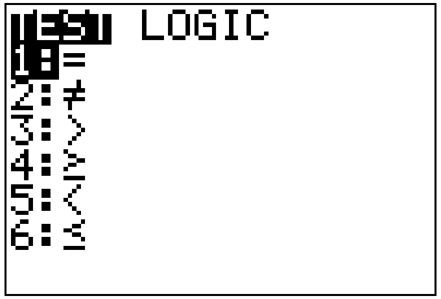


```
PROGRAM:QUAD1
:Prompt A,B,C
:Disp "USER ENTE
RED: ",A,B,C
:If A=0
:B²-4AC→D
:(-B+√(D))/(2A)→
E
```

ALPHA MATH 2nd

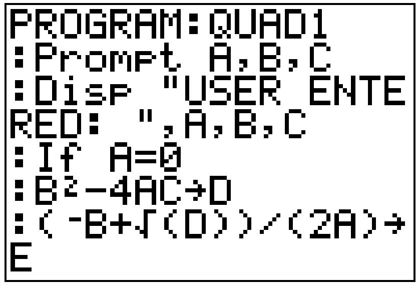
MATH ENTER 0



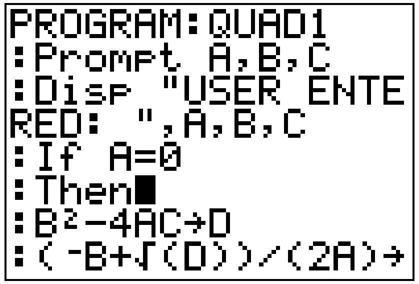


```
TEST LOGIC
1:=
2:≠
3:>
4:≥
5:<
6:≤
```

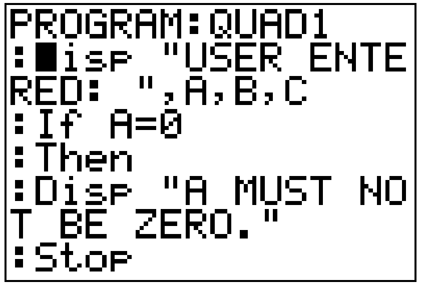
Version 3 of the QUAD1 Program



```
PROGRAM:QUAD1
:Prompt A,B,C
:Disp "USER ENTE
RED: ",A,B,C
:If A=0
:B²-4AC→D
:(-B+√(D))/(2A)→
E
```



```
PROGRAM:QUAD1
:Prompt A,B,C
:Disp "USER ENTE
RED: ",A,B,C
:If A=0
:Then█
:B²-4AC→D
:(-B+√(D))/(2A)→
```

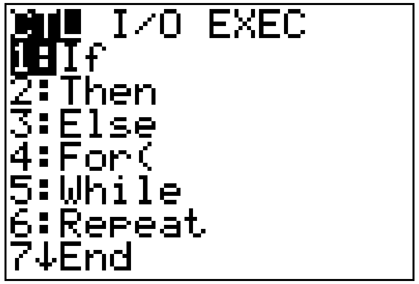


```
PROGRAM:QUAD1
:█isp "USER ENTE
RED: ",A,B,C
:If A=0
:Then
:Disp "A MUST NO
T BE ZERO."
:Stop
```

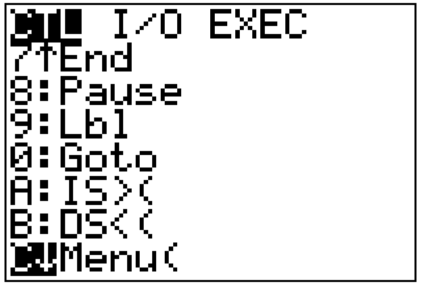
PRGM ENTER

Add the Then statement from the CTL menu. Now add a message to the user and issue a STOP command. The STOP command is also in the CTL menu, you will need to scroll down the menu to select it.

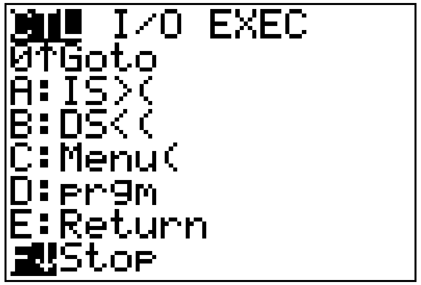
```
Prompt the user for A, B, C
If A = 0
   then
          tell the user that 0 for A is not valid
          stop the program
  else
        the value for A is not 0,
        calculate D, E, and F
        tell the user that the roots are E and F.
```



```
CTL I/O EXEC
1:If
2:Then
3:Else
4:For(
5:While
6:Repeat
7↓End
```



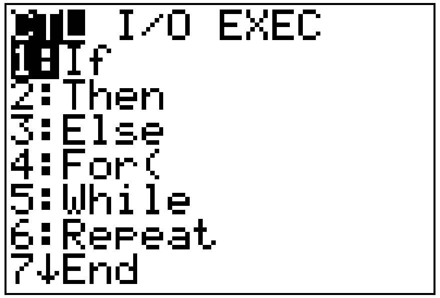
```
CTL I/O EXEC
7↑End
8:Pause
9:Lbl
0:Goto
A:IS>(
B:DS<(
C:Menu(
```



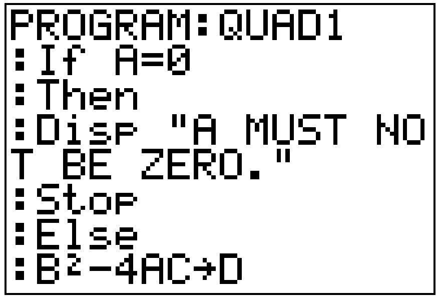
```
CTL I/O EXEC
0↑Goto
A:IS>(
B:DS<(
C:Menu(
D:prgm
E:Return
F:Stop
```

Version 3 of the QUAD1 Program

```
Prompt the user for A, B, C
If A = 0
   then
         tell the user that 0 for A is not valid
         stop the program
  else
       the value for A is not 0,
       calculate D, E, and F
       tell the user that the roots are E and F.
```



```
CTL I/O EXEC
1:If
2:Then
3:Else
4:For(
5:While
6:Repeat
7↓End
```



```
PROGRAM:QUAD1
:If A=0
:Then
:Disp "A MUST NO
T BE ZERO."
:Stop
:Else
:B²-4AC→D
```

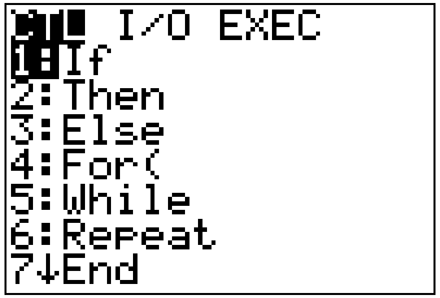
After the Stop command, and an Else command from the CTL menu. The Else statement does two things:

1) it ends the group of statements that are executed if $A = 0$

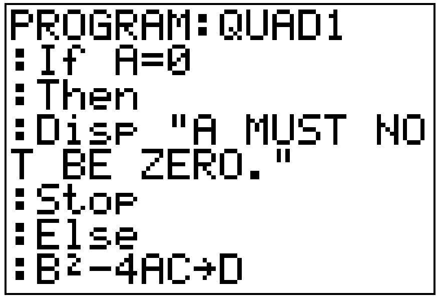
2) it begins the group of statements that are executed if A is NOT equal to 0.

Version 3 of the QUAD1 Program

```
Prompt the user for A, B, C
If A = 0
  then
      tell the user that 0 for A is not valid
      stop the program
 else
     the value for A is not 0,
     calculate D, E, and F
     tell the user that the roots are E and F.
```

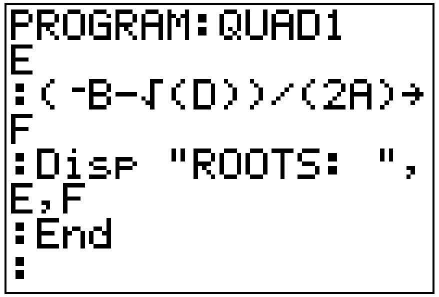


```
CTL I/O EXEC
1:If
2:Then
3:Else
4:For(
5:While
6:Repeat
7↓End
```



```
PROGRAM:QUAD1
:If A=0
:Then
:Disp "A MUST NO
T BE ZERO."
:Stop
:Else
:B²-4AC→D
```

The last thing to do is to add and End command at the end of the program - this will close off the group of statements started by the Else command.



```
PROGRAM:QUAD1
E
:(⁻B-√(D))/(2A)→
F
:Disp "ROOTS: ",
E,F
:End
:
```

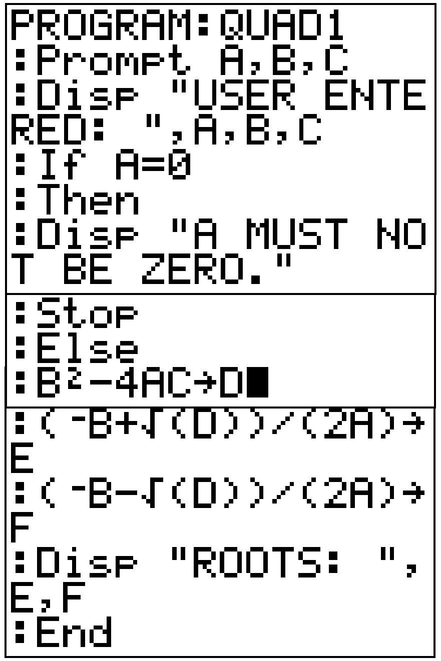
Notes:

1) Using indenting when programming makes the program more readable - but this isn't practical for the calculator.

2) Because the program would Stop as part of the If A = 0 block the End statement could have been used after the Stop command to close the If block because the rest of the program implies that the program didn't stop because A is not equal to 0.

Version 3 of the QUAD1 Program

```
Prompt A, B, C
Disp "USER ENTERED: ", A, B, C
 If A = Ø
   Then
          Disp "A MUST NOT BE ZERO."
          Stop
   Else
         B² - 4AC -> D
         (-B+√(D))/(2A)→ E
         (-B-√(D))/(2A)→ F
         Disp "ROOTS: ", E, F
   End
```



```
PROGRAM:QUAD1
:Prompt A,B,C
:Disp "USER ENTE
RED: ",A,B,C
:If A=0
:Then
:Disp "A MUST NO
T BE ZERO."
:Stop
:Else
:B²-4AC→D█
:(-B+√(D))/(2A)→
E
:(-B-√(D))/(2A)→
F
:Disp "ROOTS: ",
E,F
:End
```