

# Programming a TI-83/83+

Peter McIntyre

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>The basic program</b>	<b>1</b>
2.1	Writing the basic program . . . . .	1
2.2	Making corrections . . . . .	2
2.3	Making the program more user-friendly . . . . .	2
<b>3</b>	<b>Looping</b>	<b>3</b>
3.1	Looping for input . . . . .	3
3.2	Looping to check input . . . . .	3
3.3	Including an error message . . . . .	4
3.4	Looping through a range of input values . . . . .	5
<b>4</b>	<b>Displaying results as a table</b>	<b>6</b>
<b>5</b>	<b>Displaying results as a graph</b>	<b>7</b>
5.1	Graphing data points from a program . . . . .	7
5.2	Graphing functions from a program . . . . .	8
5.3	Putting text on a graph . . . . .	9
<b>6</b>	<b>Simple animation</b>	<b>11</b>
<b>7</b>	<b>Extensions</b>	<b>12</b>
<b>8</b>	<b>Exercises</b>	<b>14</b>
<b>9</b>	<b>Copying a program</b>	<b>15</b>
<b>10</b>	<b>Linking with a computer</b>	<b>15</b>
<b>11</b>	<b>Resources</b>	<b>16</b>
11.1	Introduction to Programming a TI-83 . . . . .	16
11.2	Sources of TI-83 programs . . . . .	16
<b>12</b>	<b>Possible solutions to the exercises</b>	<b>17</b>
<b>13</b>	<b>Keystrokes used in the programs</b>	<b>19</b>

## Contact for more help

Peter McIntyre

School of Physical, Environmental  
and Mathematical Sciences

University College (UNSW)  
Australian Defence Force Academy  
Canberra ACT 2600

Email: [p.mcintyre@adfa.edu.au](mailto:p.mcintyre@adfa.edu.au)

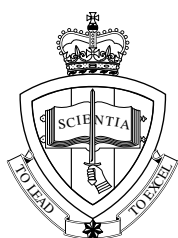
Phone: (02) 6268 8896

FAX: (02) 6268 8786

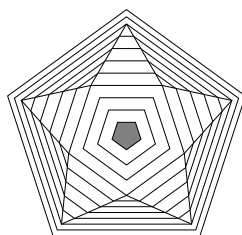
At [www.unsw.adfa.edu.au/pems/news/high\\_school/hsc\\_activities.html](http://www.unsw.adfa.edu.au/pems/news/high_school/hsc_activities.html)

- A variety of graphics-calculator activities for Years 9 and 10 — written as part of the CQTP Program.
- TI-83 programs and program information.
- *Using the TI-83/83+* — an introduction to the basic operations, suitable for Years 8–12.
- *The Graphics Screen and Accuracy* — information to help you understand the graphical and numerical limitations of a graphics calculator.
- *Coordinate Geometry on a TI-83/83+* — basic commands and a variety of problems, suitable for Years 9 and 10.
- *Population Modelling* — a variety of problems from simple exponential growth to Leslie matrices and difference equations, covering Years 7–12.
- *Calculus on a TI-83/83+* — basic commands and a variety of problems, suitable for Years 11 and 12.
- *Matrices on a TI-83/83+* — basic commands and a variety of problems, suitable for Years 11 and 12.
- *Sequences and Series on a TI-83/83+* — basic commands and a variety of problems, suitable for Years 11 and 12.
- *Complex Numbers on a TI-83/83+* — suitable for Years 11 and 12.
- *Introduction to Complex Numbers* — complex numbers from the beginning, covering the basic operations, but set in the context of complex numbers as a mathematical structure.

*These materials may be printed and copied for use by teachers and students for non-commercial educational purposes only.*



ADFA



CMA



T<sup>3</sup>

## 1 Introduction

A basic program structure is `INPUT` → `CALCULATION` → `OUTPUT`. The input is usually numbers (it might also be a function), the output can be text (e.g. ‘Complex Roots’), numbers, a graph or a table.

The programming language of the TI-83 is a subset of the BASIC programming language, containing most of the useful features of that language together with graphics commands.

Most of the commands have to be selected from a menu — you cannot type them in letter by letter. The PRGM menu selected from the home screen allows you to run, edit and create programs, while PRGM selected while editing a program gives access to most of the programming commands.

The hardest thing to start with is finding the appropriate commands on the keyboard and in menus.<sup>1</sup> *On pages 19 and 20 of these notes is a list of key strokes for the commands used in the programs here.* The Table of Functions at the back of the TI-83 Guidebook is also very useful.

## 2 The basic program

We start with a simple example and successively add features.

*Input the radius of a circle, calculate and output its area.*

### 2.1 Writing the basic program

Press the `PRGM` key and use the left arrow to move to NEW. Press `ENTER`.

Enter the name of the program (up to 8 characters) — CIRCLE. Note that the calculator is in ALPHA mode.

Press `ENTER` and you are now ready to enter the program.

Press `PRGM` and have a look at the commands in each of the sub-menus CTL (Control), I/O (Input/Output) and EXEC (used to insert the program name of a subroutine).

Here is our basic program. Press `ENTER` at the end of each line to go to the next line. Press `CLEAR` in a menu to return to the program.

Prompt R    `PRGM` I/O `2` `ALPHA` `R`  
 $\pi R^2 \rightarrow A$     `2nd` `∧` `ALPHA` `R` `x2` `STO` `ALPHA` `A`  
Disp A        `PRGM` I/O `3` `ALPHA` `A`

```
PROGRAM:CIRCLE
:Promet R
:πR2→A
:Disp A
```

When you are happy with the program, press `2nd` `QUIT` to return to the main screen.

Press `PRGM`, select the CIRCLE program by typing its number or with the cursor and `ENTER`. This displays *prgmCIRCLE* on the home screen.

---

<sup>1</sup>If you're really stumped, all the commands are in the CATALOG menu (`2nd` `0`): press a letter key (*don't* press `ALPHA` first) to go to commands starting with that letter. You can scroll upwards from the top or down to the bottom to get to the commands or characters that don't contain letters.



## 3 Looping

### 3.1 Looping for input

Maybe we'd like to enter a number of values for  $R$  without having to rerun the program each time. We do this with a simple loop using a *Label* and a *Goto*.

```
ClrHome
Disp "AREA OF A CIRCLE"
Lbl 1      [PRGM] [9] [1]
Input "RADIUS:␣", R
 $\pi R^2 \rightarrow A$ 
Disp "AREA", A
Goto 1     [PRGM] [0] [1]
```

This makes entering a number of radius values easier, but how do we stop the program? Press **[ON]** and select *Quit*. This stops any program and returns to the home screen. Use **[ON]** *Goto* to go back to edit the program — useful when developing a program.

### 3.2 Looping to check input

Another place for a loop is to test whether an input value falls within a specified range. In our example, we expect  $R \geq 0$  (although it doesn't matter in this case because we square  $R$ ). Let's use a loop to test for this. We also introduce the *If* statement here.

```
ClrHome
Disp "AREA OF A CIRCLE"
Lbl 1
Input "RADIUS:␣", R
If R < 0      If is [PRGM] [1]; < is [2nd] [TEST] (MATH key) [5]
Goto 1       Do this step if the 'If' is true, otherwise skip this step
 $\pi R^2 \rightarrow A$ 
Disp "AREA", A
Goto 1
```

Check your program with suitable inputs, i.e. at least one negative  $R$ .

See Extension 1 on page 12 for another way to use such a loop and Extension 2 on page 12 for a more elegant error check. Make a copy of your program (see Section 9) so you can come back to these extensions later.

### 3.3 Including an error message

Let's say we want to tell the user that he or she has made a mistake in the input. Such diagnostics are always useful, especially for students.

Here we embellish the *If* statement with a *Then* and *End*, and introduce *Pause*, which stops the program until the user presses `ENTER`.

That a program has Paused is indicated by shimmering dots in the top right corner of the screen.

Note that anything in inverted commas is text, and can be whatever you want.

```

ClrHome

Disp "AREA OF A CIRCLE"

Lbl 1

Input "RADIUS:␣", R

If R < 0
  Then    PRGM 2
  Disp "****MAKE R≥0****" * is ⊗ — used for emphasis here
  Pause   PRGM 8
Goto 1
  End    PRGM 7 — ends the If Then
 $\pi R^2 \rightarrow A$ 
Disp "AREA", A
Goto 1

```

Don't forget suitable tests for this version of the program.

### 3.4 Looping through a range of input values

Here we use a *For* loop to carry out the calculations for a specified range of (equally spaced) input values. The *For* command has the format

*For* (*R*, *F*, *L*, *S*)

*commands*

*End*

*R* starts with a value of *F* and executes *commands*. *R* is then increased<sup>2</sup> by *S* and *commands* are executed again. This continues until *R* is greater than or equal to *L*.

Remove the lines (use `CLEAR` `DEL`) that test for  $R < 0$  to keep the program simple.

ClrHome

Disp "AREA OF A CIRCLE"

Input "FIRST R: ", F      *prompt for the initial value of R*

Input "LAST R: ", L      *prompt for the final value of R*

Input "R STEP: ", S      *prompt for the increment in R*

For (R, F, L, S)      `PRGM` `4`

$\pi R^2 \rightarrow A$

Disp "R, AREA", R, A

Pause      *pause to read answer*

End      *end For loop*

---

<sup>2</sup>S can be negative, so that *R* is decreased. In this case, *L* must be less than *F*, and the program stops when  $R \leq L$ .

## 4 Displaying results as a table

Rather than displaying the areas one by one as we did in the previous example, we can put them in a table. In a table, we can scroll up and down to see all the answers.

To make up a table, we need to put the R and A values in lists. Here we use list L1 for R and list L2 for A.

Lists L1 – L6 are built in to the calculator and can be accessed from the keyboard. You can also give lists names.

The variable I in the following program is the list index — we have to increment it by 1 each time we store a value in a list.

ClrHome

SetUpEditor      **[STAT]** **[5]**      *ensures that lists L1 – L6 are shown when we press [STAT] EDIT after running the program*

ClrList L1, L2      **[STAT]** **[4]** **[2nd]** **[1]** **[↓]** **[2nd]** **[2]**

1 → I      *initialise the list index*

Disp "AREA OF A CIRCLE"

Input "FIRST R: ", F

Input "LAST R: ", L

Input "R STEP: ", S

For (R, F, L, S)

R → L1(I)      L1 is **[2nd]** **[1]**

$\pi R^2$  → L2(I)      L2 is **[2nd]** **[2]**

I + 1 → I      *increment the list index by 1*

End

Disp "", "PRESS **[STAT]** **[EDIT]**"      *tell the user how to view the answers — the inverted commas with no enclosed characters give a blank line in the output*

Run the program with R varying from 0 to 10 in steps of 1 and look at the results. Press **[2nd]** **[QUIT]** to return to the home screen from the table.



## 5.2 Graphing functions from a program

Graphing a function from a program is simple — the function has to be defined in  $\boxed{Y=}$ , either before the program is run or from within the program. Be careful in a program that the right function is turned on — use `FnOn` and `FnOff` to do this. If you define a function within a program, it is automatically turned on.

There are several commands you can use in a program to display a graph. All of these will cause any functions *and any plots* turned on to be graphed.

- `TRACE` — displays the graph with a cursor you can move along the functions/points plotted, just as you would using `TRACE` manually. `TRACE` pauses the program — press  $\boxed{\text{ENTER}}$  to continue.
- `Input` —  $\boxed{\text{PRGM}}$  I/O  $\boxed{1}$  with no arguments. *Input* displays the graph with a cursor that you can move around the screen. *Input* pauses the program — press  $\boxed{\text{ENTER}}$  to continue.
- `DispGraph` —  $\boxed{\text{PRGM}}$  I/O  $\boxed{4}$ . *DispGraph* displays the graph without a cursor. *DispGraph* does not pause the program — if it is not the last statement in the program, you will have to follow it with a *Pause*.
- `DrawF` —  $\boxed{\text{DRAW}}$   $\boxed{6}$ . *DrawF* followed by a function definition either explicitly in terms of  $X$  (e.g.  $3X + 4$ ) or one of  $Y_1 - Y_0$ , draws that function. Use *Input* or *Pause* to view the graph. A function drawn with *DrawF* cannot be `TRACEd`. An advantage of *DrawF* is that it allows successive graphs to be superimposed — see Exercise 1.

To continue our example and do some curve fitting, let's fit a quadratic function to our data points, store the fitted function in  $Y_1$  and graph it together with the data. We expect a good fit! The data-fitting commands are in the  $\boxed{\text{STAT}}$  `CALC` menu.

`ClrHome`

`FnOff`

`PlotsOff`

`ClrDraw`

`0 → Xscl`

`0 → Yscl`

`ClrList L1, L2`

`1 → I`

`Disp "AREA OF A CIRCLE"`

`Input "FIRST R: ", F`

`Input "LAST R: ", L`

`Input "R STEP: ", S`

```

For (R, F, L, S)
R → L1 (I)
πR2 → L2 (I)
I + 1 → I
End
Plot1 (Scatter, L1, L2, □)
ZoomStat
Trace      view the data points
QuadReg L1, L2, Y1  [STAT] CALC [5]; Y1 is [VARS] Y-VARS [1] [1]
Trace      view the data points and the fitted curve
Disp "", "PRESS STAT EDIT"

```

Run this program and see if you get the expected fit to the data points. Press `[ENTER]` after the points are plotted to see the fitted curve. Then try the up and down arrows as well as the left and right arrows. Press `[ENTER]` to finish the program and look at Y1 in `[Y=]`.

### 5.3 Putting text on a graph

Sometimes it is useful to annotate graphs so that the user can understand what is being plotted or be prompted for the next key stroke(s).

As an example, in the graph of our last two versions of the program, the calculator pauses in TRACE. Possible keys to press are the arrow keys to move along or between the graphs and `[ENTER]` to finish the program. We use the *Text* command to indicate this on the graph.

The *Text* command is of the form *Text*(*row*, *column*, "*text*" or *variable*). The coordinates *row* and *column* are pixel coordinates rather than the coordinates given by the WINDOW: *row* varies from 0 (top of screen) to 57 (bottom of screen), while *column* varies from 1 (left of screen) to 94 (right of screen). It is usually necessary to experiment with the pixel coordinates to get the text in the right place.

*Text* can also be used to display the value of a variable — put the variable name in the command without inverted commas.

Any combination of text and variable values can be put together in a *Text* command — just put them one after the other, separated by commas. This is useful if you want to display the name of a parameter and its value: *Text*(0, 2, "N=", N).

```

ClrHome

FnOff

PlotsOff

ClrDraw

0 → Xscl

0 → Yscl

ClrList L1, L2

1 → I

Disp "AREA OF A CIRCLE"

Input "FIRST R:␣", F

Input "LAST R:␣", L

Input "R STEP:␣", S

For (R, F, L, S)

R → L1(I)

 $\pi R^2$  → L2(I)

I + 1 → I

End

Plot1(Scatter, L1, L2, □)

ZoomStat

Text(8, 13, "[ARROWS]␣␣␣[ENTER]")  DRAW 0 ; [ is 2nd ×, ] 2nd −

Trace

QuadReg L1, L2, Y1

Text(8, 13, "[ARROWS]␣␣␣[ENTER]")

Trace

Disp "", "PRESS STAT EDIT"

```

## 6 Simple animation

Animation is achieved by creating the separate frames, storing them in the picture memories Pic1, Pic2, . . . , Pic0 and then recalling them one by one.

This program also illustrates the use of *getkey* to receive input from the keyboard while a program (loop) is running — here we use the Q key to stop the program.<sup>3</sup>

### Program ANIMATE

```
FnOff
PlotsOff
AxesOff      [2nd] [ZOOM] (FORMAT); use the cursor and [ENTER] to select
ZStandard    [ZOOM] [6] — standard axes
ZSquare      [ZOOM] [5] — same scale on each axis
ClrDraw      clear the graphics screen
Shade( $-\sqrt{64 - X^2}$ ,  $\sqrt{64 - X^2}$ ) [DRAW] shade between the two functions
Text(1, 1, "Q:␣QUIT") on-screen prompt to tell user how to stop the program
StorePic 1    [DRAW] STO menu store the first screen in Pic1
ClrDraw      clear the graphics screen
Circle(0, 0, 8) [DRAW] menu circle centre (0, 0), radius 8
Text(1, 1, "Q:␣QUIT")
StorePic 2    store the second screen in Pic2
Lbl 1        start animation loop
ClrDraw      clear the graphics screen
RecallPic 1   [DRAW] STO menu recall the first screen
For(I, 1, 150) delay loop to view screen
End
ClrDraw      clear the graphics screen
RecallPic 2   recall the second screen
For(I, 1, 120) shorter delay loop to allow for getkey test
End
If getKey=74  [PRGM] I/O menu — getkey detects if a key has been pressed
Goto 2       if the Q key (key 74) has been pressed, go to Label 2 (tidy up and quit)
Goto 1       if the Q key hasn't been pressed, go to Label 1 (start of loop)
Lbl 2        tidy up
AxesOn
DelVar Pic1   [PRGM] (scroll up) [VARS] [4] [1]
DelVar Pic2
ClrHome
```

---

<sup>3</sup>[ON] Quit would stop it too, but there would be no ‘tidying up’ as here.

## 7 Extensions

### Extension 1

Follows on from the example on page 3.

We could also use  $R < 0$  as a way to stop the program, rather than using `ON Quit`. Note the message telling the user this after the main heading.

```
ClrHome
Disp "AREA OF A CIRCLE", "R<0 TO STOP"
Lbl 1
Input "RADIUS:", R
If R < 0
  Stop    PRGM F or scroll up and select
 $\pi R^2 \rightarrow A$ 
Disp "AREA", A
Goto 1
```

### Extension 2

Follows on from the example on page 3.

For a simpler, more elegant check on the input, we can use a *Repeat* loop.

```
ClrHome
Disp "AREA OF A CIRCLE"
Lbl 1
Repeat R ≥ 0    PRGM 6 — repeat down to End until the condition is true
Input "RADIUS:", R
End
 $\pi R^2 \rightarrow A$ 
Disp "AREA", A
Goto 1
```

### Extension 3

Here we introduce a custom *Menu* in a program that calculates the area of a triangle, rectangle or circle. Only an outline of the program is given here — the details are to be completed in Exercise 2.

*Menu*( is in the **PRGM** CTL menu. *Output*(, in the **PRGM** I/O menu, is like *Text*, but writes on the home screen rather than the graphics screen.

Note the use of a suggestive letter for each label.

#### Program AREA

Lbl M

ClrHome

Menu ("▣AREA▣OF▣A . . .▣", "TRIANGLE", T, "RECTANGLE", R, "CIRCLE", C, "QUIT", Q)

Lbl T

*Insert prompts, inputs and calculations to calculate the area of a triangle.  
Store the area in A.*

Goto D

Lbl R

*Insert prompts, inputs and calculations to calculate area of rectangle.  
Store the area in A.*

Goto D

Lbl C

*Insert prompts, inputs and calculations to calculate area of circle.  
Store the area in A.*

Lbl D

Disp "AREA", A

Output (8, 10, "[ENTER]")      *tells the user to press **ENTER** to continue*

Pause

Goto M

Lbl Q

ClrHome

## 8 Exercises

If these exercises seem too hard, think up a suitable calculation and program it.

### Exercise 1

Assume that  $Y_1 = AX^2 + B$  and that a suitable WINDOW has been set. Your program called FAMILY should

- clear the home screen
- turn off all functions
- turn off stat plots
- clear the graphics screen
- display a header on the home screen
- start a loop (*Lbl*)
- input values for A and B
- draw the graph (*DrawF Y<sub>1</sub>*)
- pause to allow you to look at the graph
- return for another set of values for A and B (end of loop)

This program allows you to see the effect on the function of changing the parameters A and B.

Use a WINDOW of  $[-5, 5, 1] \times [-10, 10, 2]$ . Try  $A = 1, -1, 2, 0.5$  with  $B = 0$ . Use ON *Quit* to stop the program and rerun with  $A = 1$  and  $B = 0, \pm 2, \pm 5$ .

You can put in  $Y_1$  any function containing two parameters A and B, and plot the corresponding family of curves using FAMILY. Try, for example,  $Y_1 = A(X-B)^2$ .

### Exercise 2

Write a user-friendly program that calculates the area of a triangle, rectangle or circle, given appropriate inputs. *Hint*: see Extension 3.

### Exercise 3

Find the real roots of the quadratic equation  $Ax^2 + Bx + C = 0$ , where  $A$ ,  $B$  and  $C$  are inputs. Display a message if the roots are complex (later you might like to extend the program to complex roots). Finally plot the quadratic: specify the  $x$  range in the program ( $X_{\min}$ ,  $X_{\max}$ ) — the roots should help here — and use *ZoomFit*, which calculates a suitable  $y$  range and plots the graph.

To put the quadratic function in  $Y_1$ : " $AX^2 + BX + C$ "  $\rightarrow Y_1$ . This works both within a program and from the home screen (although entering the function using Y= is clearly an easier option in the second case).

## 9 Copying a program

To make a copy of a program or to change the name of a program, first create a new program (`PRGM NEW`) with a suitable (different) name.

Press `2nd RCL` (on the `STO` key), then `PRGM EXEC` and select the program you wish to copy or rename by number or with the cursor and `ENTER`. *Rcl* and the name of the program should appear at the bottom of the screen.

Press `ENTER` to recall the program. (You can use this process to insert one program into another as well. Go to where you wish to insert and carry out the above process.)

You can now edit the new program or exit using `2nd QUIT`.

Delete the old program using `2nd MEM 2` if you are just changing its name.

## 10 Linking with a computer

The TI-83 can be linked to a computer using a special cable (which has to be bought) and software (which comes with the cable or is free on the web). The latest version of the software, TI CONNECT, is available at

*[education.ti.com/us/product/accessory/connectivity/features/software.html](http://education.ti.com/us/product/accessory/connectivity/features/software.html)*

Click on *downloads* on the left to download.

You can use the software to copy programs from the calculator to the computer for storage or printing, and to ‘grab’ a screen image, which you can then print or store as a file to put in notes (very useful).

## 11 Resources

### 11.1 Introduction to Programming a TI-83

These Texas Instruments notes are available at [education.ti.com/downloads/guidebooks/eng/ti83progguide.pdf](http://education.ti.com/downloads/guidebooks/eng/ti83progguide.pdf).

### 11.2 Sources of TI-83 programs

The TI-83 Guidebook contains a number of programs in the Activities chapter which illustrate many of the possibilities.

The TI web site [education.ti.com/global/archreadme.html](http://education.ti.com/global/archreadme.html) contains an extensive list of programs for TI calculators. You download these programs from the web using your web browser and then into your calculator with TI GRAPHLINK or TI CONNECT.

[www.ticalc.org](http://www.ticalc.org) is another site that contains programs, as well as lots of other useful information about TI calculators.

The quality of programs varies enormously on both these sites, but you will find plenty of ideas of what is possible, even if the programs don't work too well.

At our ADFA web site [www.unsw.adfa.edu.au/pems/news/high\\_school/hsc\\_activities.html](http://www.unsw.adfa.edu.au/pems/news/high_school/hsc_activities.html), we have a number of TI-83 programs (with instructions) used in our Calculus, Linear Algebra, Probability and Discrete Dynamics/Epidemiology courses. A number of these programs are suitable for College courses.

## 12 Possible solutions to the exercises

### Exercise 1

#### Program FAMILY

```
ClrHome
ClrDraw
FnOff
PlotsOff
Disp "FAMILY PROGRAM", "PARAMETERS A,B"
Lbl 1
Prompt A,B
DrawF Y1
Input
Goto 1
```

### Exercise 2

#### Program AREA

```
Lbl M
ClrHome
Menu (" AREA OF A... ", "TRIANGLE", T, "RECTANGLE", R, "CIRCLE", C, "QUIT", Q)
Lbl T
Disp " TRIANGLE"
Input "BASE:", B
Input "HEIGHT:", H
0.5BH → A
Goto D
Lbl R
Disp " RECTANGLE"
Input "LENGTH:", L
Input "WIDTH:", W
LW → A
Goto D
Lbl C
Disp " CIRCLE"
Input "RADIUS:", R
 $\pi R^2 \rightarrow A$ 
Lbl D
Disp "AREA", A
Output (8, 10, "[ENTER]")
Pause
Goto M
Lbl Q
ClrHome
```

### Exercise 3

#### Program QUAD

```
ClrHome
ClrDraw
FnOff
PlotsOff
"AX2 + BX + C" → Y1
Disp "□□QUADRATIC□EQN", "□□□AX2 + BX + C = 0"
Prompt A, B, C
B2 - 4AC → D
If D < 0
Then
Disp "COMPLEX□ROOTS"
Stop
End
(-B+√(D))/(2A) → R
(-B-√(D))/(2A) → S
Disp "ROOTS", R, S
Pause
If R < S
Then
R - 2 → Xmin
S + 2 → Xmax
Else
S - 2 → Xmin
R + 2 → Xmax
End
0 → Xscl
ZoomFit
```

Once the program has finished, you can press TRACE or change the WINDOW and re-graph.

## 13 Keystrokes used in the programs

□	<code>2nd</code> <code>STATPLOT</code> ( <code>Y=</code> key) <code>MARK</code> <code>1</code>
[	<code>2nd</code> <code>×</code>
]	<code>2nd</code> <code>-</code>
:	<code>ALPHA</code> <code>□</code>
"	<code>ALPHA</code> <code>+</code>
=	<code>2nd</code> <code>TEST</code> (MATH key) <code>1</code>
<	<code>2nd</code> <code>TEST</code> (MATH key) <code>5</code>
$\pi$	<code>2nd</code> <code>∧</code>
□ (space)	<code>ALPHA</code> <code>0</code>
$\sqrt{}$	<code>2nd</code> <code>x<sup>2</sup></code>
→	<code>STO</code>
AxesOff	<code>2nd</code> <code>FORMAT</code> (ZOOM key) — use the cursor and <code>ENTER</code> to select
Circle(	<code>2nd</code> <code>DRAW</code> (PRGM key) <code>9</code>
ClrDraw	<code>2nd</code> <code>DRAW</code> (PRGM key) <code>1</code>
ClrHome	<code>PRGM</code> I/O <code>8</code>
ClrList	<code>STAT</code> <code>4</code>
<i>colon</i>	<code>ALPHA</code> <code>□</code>
DelVar	<code>PRGM</code> <code>ALPHA</code> <code>G</code> or scroll up and select
Disp	<code>PRGM</code> I/O <code>3</code>
DrawF	<code>2nd</code> <code>DRAW</code> (PRGM key) <code>6</code>
Else	<code>PRGM</code> <code>3</code>
End	<code>PRGM</code> <code>7</code>
FnOff	<code>VARS</code> Y-VARS <code>4</code> <code>2</code>
For(	<code>PRGM</code> <code>4</code>
getKey	<code>PRGM</code> I/O <code>7</code>
Goto	<code>PRGM</code> <code>0</code>
If	<code>PRGM</code> <code>1</code>
Input	<code>PRGM</code> I/O <code>1</code>
L1	<code>2nd</code> <code>1</code>
L2	<code>2nd</code> <code>2</code>

Lbl	<b>PRGM</b> <b>9</b>
Menu(	<b>PRGM</b> <b>ALPHA</b> <b>C</b> or scroll up and select
Output(	<b>PRGM</b> I/O <b>6</b>
Pause	<b>PRGM</b> <b>8</b>
PlotsOff	<b>2nd</b> <b>STATPLOT</b> (Y= key) <b>4</b>
Prompt	<b>PRGM</b> I/O <b>2</b>
Plot1(	<b>2nd</b> <b>STATPLOT</b> (Y= key) <b>1</b>
QuadReg	<b>STAT</b> <b>CALC</b> <b>5</b>
RecallPic	<b>2nd</b> <b>DRAW</b> (PRGM key) <b>STO</b> <b>2</b>
Repeat	<b>PRGM</b> <b>6</b>
Scatter	<b>2nd</b> <b>STATPLOT</b> (Y= key) <b>TYPE</b> <b>1</b>
SetUpEditor	<b>STAT</b> <b>5</b>
Shade(	<b>2nd</b> <b>DRAW</b> (PRGM key) <b>7</b>
<i>space</i>	<b>ALPHA</b> <b>0</b>
Stop	<b>PRGM</b> <b>F</b> or scroll up and select
StorePic	<b>2nd</b> <b>DRAW</b> (PRGM key) <b>STO</b> <b>1</b>
Text(	<b>2nd</b> <b>DRAW</b> (PRGM key) <b>0</b>
Then	<b>PRGM</b> <b>2</b>
Xscl	<b>VARS</b> <b>1</b> <b>3</b>
Y1	<b>VARS</b> <b>Y-VARS</b> <b>1</b> <b>1</b>
Yscl	<b>VARS</b> <b>1</b> <b>6</b>
ZoomFit	<b>ZOOM</b> <b>0</b>
ZoomStat	<b>ZOOM</b> <b>9</b>
ZSquare	<b>ZOOM</b> <b>5</b>
ZStandard	<b>ZOOM</b> <b>6</b>