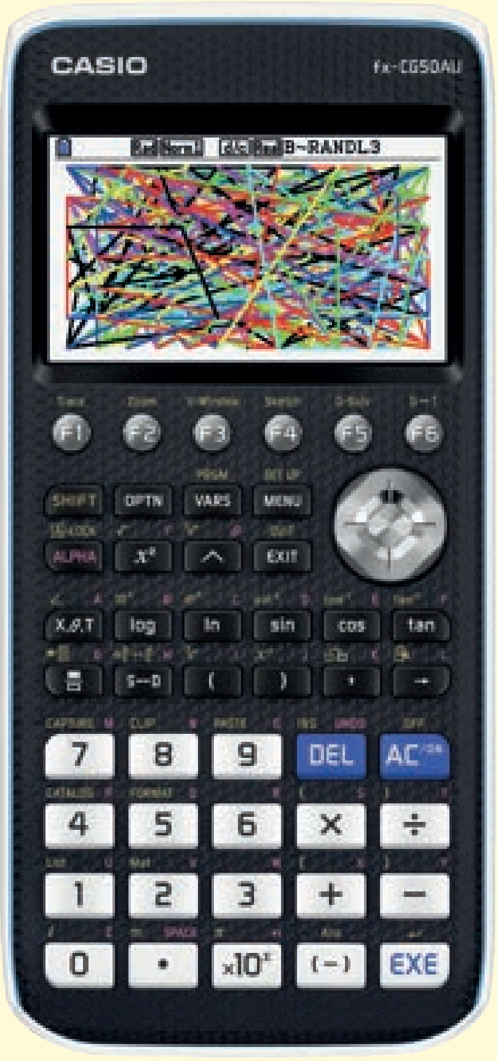
How Do I CODE on a CASIO fx-CG50AU

Making it possible for beginners

Includes pre-written code that you modify in supportive exercises



```
"PRESS EXE TO START"◢
Cls↵
ViewWindow -6.3,6.3,1,-3.1,3.1,1↵
AxesOff↵
GridOff↵
CoordOff↵
LabelOff↵
S-L-Normal↵

Lbl 1↵

RanInt#(1,7)->P↵
RanInt#(-6,6)->A↵
RanInt#(-3,3)->B↵
RanInt#(-6,6)->C↵
RanInt#(-3,3)->D↵

If P=1↵
 Then Black F-Line A,B,C,D↵
IfEnd↵

If P=2↵
 Then Blue F-Line A,B,C,D↵
IfEnd↵

If P=3↵
 Then Red F-Line A,B,C,D↵
IfEnd↵

If P=4↵
 Then Magenta F-Line A,B,C,D↵
IfEnd↵

If P=5↵
 Then Green F-Line A,B,C,D↵
IfEnd↵

If P=6↵
 Then Cyan F-Line A,B,C,D↵
IfEnd↵

If P=7↵
 Then Yellow F-Line A,B,C,D↵
IfEnd↵

Goto 1↵
```

Team Steps

Made in Australia

How Do I CODE on a CASIO fx-CG50AU

1st Edition.

First published in 2019.

Questions about this publication should be directed to support@stepsinlogic.com

ISBN 978-0-9807619-8-6

Art by Taryn.

Contents

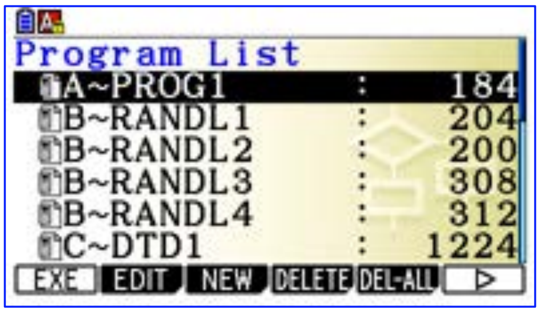
1. Before your start

In this book you will execute pre-written lines of code (programs) and modify that code to make the program do as you want it too.

Thus you should first load the pre-written programs that are referred to in this book into either your fx-CG-50AU hand-held or your fx-CG Manager Plus for CG-50 series.

1.1 Loading the programs into the fx-CG-50AU (hand-held)

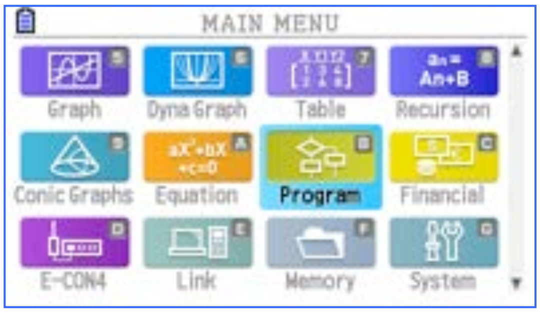
1. Visit https://casioeducation.com.au/software-emulators
2. Download the file prg4book.g3m and save it to your chosen location.
3. Connect your fx-CG-50AU to your computer via USB and choose USB Flash (F1).
4. The flash memory drive of your fx-CG-50AU will mount on your computer. Copy the file prg4book.g3m into the flash memory drive of your fx-CG-50AU. Disconnect flash memory drive of your fx-CG-50AU from your computer.
5. Open the Memory application on the fx-CG-50AU.
6. Choose Storage Memory (F2).
7. Highlight prg4book.g3m, SELECT (F1) it and then choose COPY (F2).
8. Once the copying process is complete press EXIT.
9. Open the Program application and you will see the pre-written program's file names.


```
Program List
A~PROG1      :   184
B~RANDL1     :   204
B~RANDL2     :   200
B~RANDL3     :   308
B~RANDL4     :   312
C~DTD1       :  1224
```

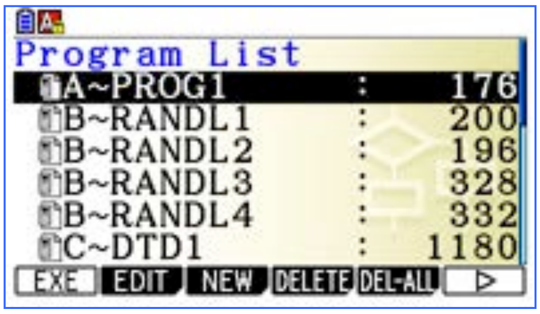
1.2 Loading the programs into the fx-CG-50 Manager Plus (software)

1. Visit https://casioeducation.com.au/software-emulators
2. Download the file prg4book.g3m and save it to your chosen location.
3. Open the Memory application on the fx-CG-50 Manager Plus.
4. Choose Import/Export (F3).
5. Choose Import (F1).
6. A "finder" window will open on your computer; locate the file prg4book.g3m.
7. Highlight prg4book.g3m and choose Open.
8. ROOT will be selected on your fx-CG-50 Manager Plus, press SAVE (F1).
9. When the save process is completed press EXIT.
10. Choose Storage Memory (F2).
11. Highlight prg4book.g3m, SELECT (F1) it and then choose COPY (F2).
12. Once the copying process is complete press EXIT.
13. Open the Program application and you will see the pre-written program's file names (as shown above.

2. The basics of a program



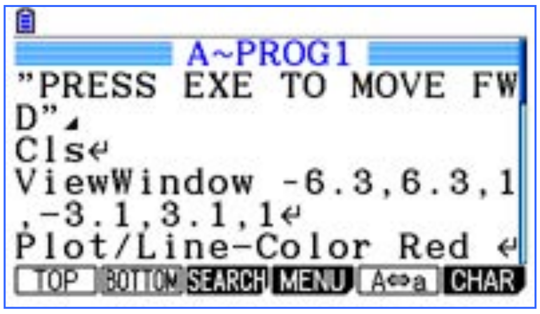
Open the program application.



```
Program List
A~PROG1      : 176
B~RANDL1     : 200
B~RANDL2     : 196
B~RANDL3     : 328
B~RANDL4     : 332
C~DTD1       : 1180
```

Use the cursor key to highlight (select) a program.

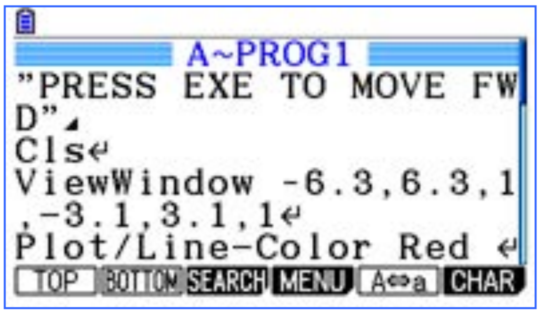
EDIT (F2) shows the lines of code in the program that has been selected.



```
A~PROG1
"PRESS EXE TO MOVE FWD"◢
Cls↵
ViewWindow -6.3,6.3,1,-3.1,3.1,1↵
Plot/Line-Color Red ↵
```

EXIT will return you to the program list.

2.1 Understanding the code of A~PROG1.


```
A~PROG1
"PRESS EXE TO MOVE FW
D"◢
Cls↵
ViewWindow -6.3,6.3,1
,-3.1,3.1,1↵
Plot/Line-Color Red ↵
```

The complete program is as follows:

```
"PRESS EXE TO MOVE FWD"◢
Cls↵
ViewWindow -6.3,6.3,1,-3.1,3.1,1↵
S-L-Normal↵

Plot/Line-Color Red↵
PlotOn -2,1◢
PlotOn -4,1◢

Plot/Line-Color Black↵
PlotOn -2,3◢
PlotOn -4,3◢

Plot/Line-Color Blue↵
F-Line 0,0,3,3◢

ClrText↵

"WHAT'S NEXT?"◢

Plot/Line-Color Magenta↵
Circle 1,1,2◢

ClrText↵
"THE END"↵
Stop↵
```

Each line of code tells the fx-CG50AU to do something when the program is executed.

You can move up and down the lines of code using the cursor keys.
TOP (F1): jumps the cursor to the top of the script.
BOTTOM (F2) jumps the cursor to the bottom of the script

When the program is executed, the first line (`"PRESS EXE TO MOVE FWD"◢`) will display a message on the screen.
All messages must be enclosed in quotes.

The symbol ◢, when at the end of a line of code, makes the fx-CG50AU pause and display the preceding object (the message in this case).

To continue, the user presses EXE.

The symbol ↵ is a carriage return to signify the end of a line of code and is automatically added by the calculator when entering code.

In this program, the calculator draw various objects on the Cartesian plane.

`ViewWindow xmin,xmax,scale,ymin,ymax,scale` 'scales' the plane.

`Plot/Line-Color colour` sets the colour of the objects that are plotted by the proceeding lines of code.

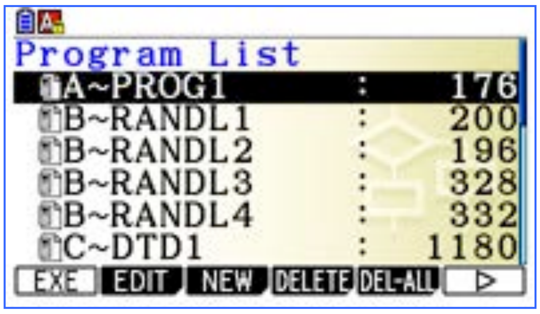
`Plot x,y` plots a *point*, at a specified location on the Cartesian plane, (-2,1) for example.

`F-Line a,b,c,d` plots a line between two specified locations on the Cartesian plane, (0,0) and (3,3) in this case.

`Circle x,y,r` plots a circle with centre at a specified location on the Cartesian plane and with specified radius, centre (1,1) and radius 2 in this case.

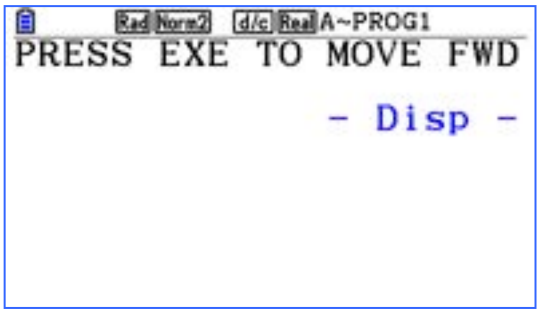
Note that the empty lines between lines of code, seen above, are permitted in the Program application, but require a carriage return. They are often not included, however.

2.2 Running the code of A~PROG1.



```
Program List
A~PROG1   :  176
B~RANDL1  :  200
B~RANDL2  :  196
B~RANDL3  :  328
B~RANDL4  :  332
C~DTD1    : 1180
EXE EDIT NEW DELETE DEL-ALL ▷
```

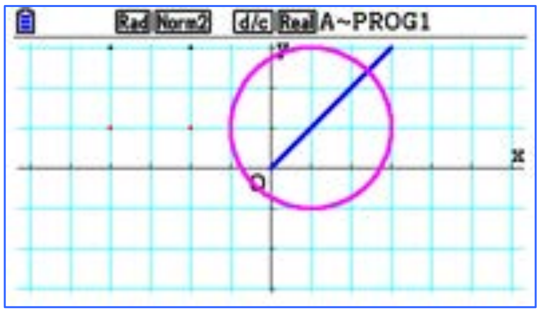
Highlight the program named A~PROG1. Press EXE.



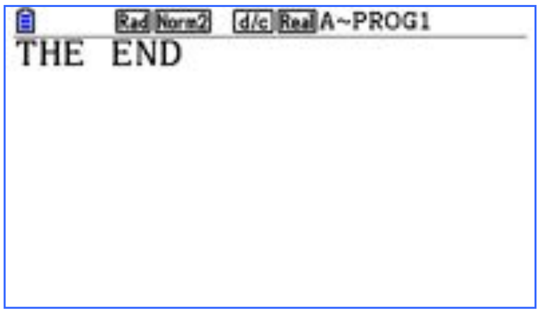
```
Rad Norm2 d/c Real A~PROG1
PRESS EXE TO MOVE FWD
                - Disp -
```

– Disp – tells us the program has paused to display something, a message in this case.

Press EXE to move the program on past this first pause.

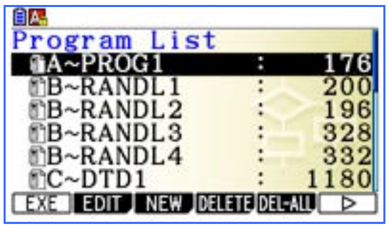


Continue to press EXE to move the program on past each display/pause.



```
Rad Norm2 d/c Real A~PROG1
THE END
```

Pressing EXIT or EXE, once the program is done, will display the “select a program” (Program List) screen.



```
Program List
A~PROG1   :  176
B~RANDL1  :  200
B~RANDL2  :  196
B~RANDL3  :  328
B~RANDL4  :  332
C~DTD1    : 1180
EXE EDIT NEW DELETE DEL-ALL ▷
```

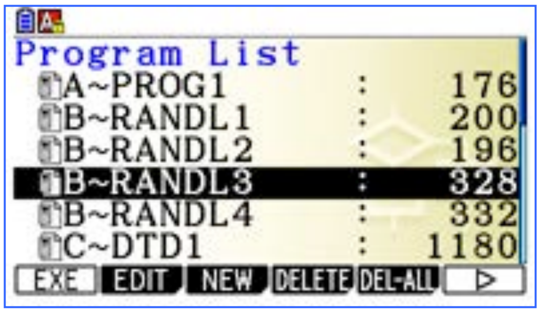
Note that the following lines of code:

- `ViewWindow -6.3,6.3,1,-3.1,3.1,1↵`
- `S-L-Normal↵`
- `Plot/Line-Color Red↵`

set the *global* behaviour of the fx-CG50AU. Thus, after running a program, if you draw a graph, for example, the ViewWindow will remain as set by the program. The ViewWindow can be changed using V-Window (SHIFT then F3). The other settings can be changed using SETUP (SHIFT then Menu).

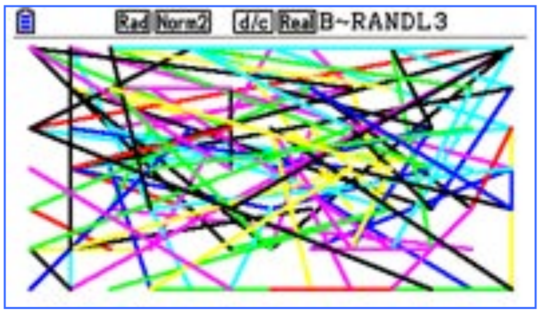
If you did not see the axes and or grid, as seen above, it will because of the personal settings on your machine. The program did not effect change on them, as it did the ViewWindow settings. You will see how to control these from within a program very soon.

3. Breaking, If-Then-IfEnd, Lbl-Goto & variables



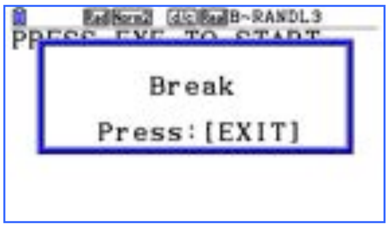
```
Program List
 A~PROG1      :   176
 B~RANDL1     :   200
 B~RANDL2     :   196
 B~RANDL3     :   328
 B~RANDL4     :   332
 C~DTD1       :  1180
```

Select the program named B~RANDL3.
Press EXE to run the program.

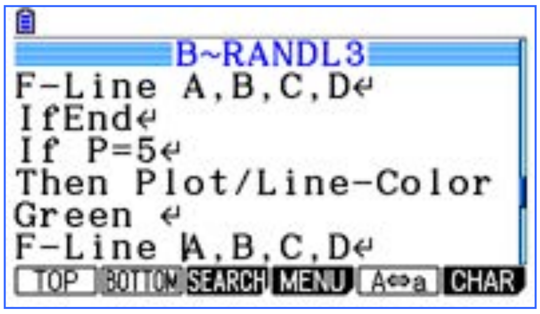


It seems to go on forever!

To break (terminate) a program, at any time – press AC/ON



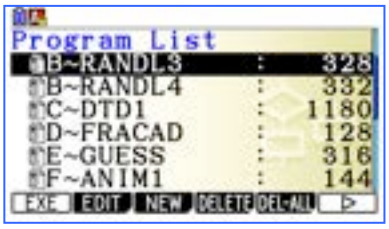
```
Break
Press:[EXIT]
```



```
B~RANDL3
F-Line A,B,C,D↵
IfEnd↵
If P=5↵
Then Plot/Line-Color
Green ↵
F-Line A,B,C,D↵
```

Press EXIT, as prompted, and the lines of code of the program are shown.

Pressing EXIT again will return you to the "select a program" (Program List) screen.



```
Program List
 B~RANDL3     :   328
 B~RANDL4     :   332
 C~DTD1       :  1180
 D~FRACAD     :   128
 E~GUESS      :   316
 F~ANIM1      :   144
```

How are all those colourful lines being produced, and how is their position determined?

```
"PRESS EXE TO START"◢
Cls↵
ViewWindow -6.3,6.3,1,-3.1,3.1,1↵
AxesOff↵
GridOff↵
CoordOff↵
LabelOff↵
S-L-Normal↵

Lbl 1↵

RanInt#(1,7)->P↵
RanInt#(-6,6)->A↵
RanInt#(-3,3)->B↵
RanInt#(-6,6)->C↵
RanInt#(-3,3)->D↵

If P=1↵
  Then Black F-Line A,B,C,D↵
IfEnd↵

If P=2↵
  Then Blue F-Line A,B,C,D↵
IfEnd↵

If P=3↵
  Then Red F-Line A,B,C,D↵
IfEnd↵

If P=4↵
  Then Magenta F-Line A,B,C,D↵
IfEnd↵

If P=5↵
  Then Green F-Line A,B,C,D↵
IfEnd↵

If P=6↵
  Then Cyan F-Line A,B,C,D↵
IfEnd↵

If P=7↵
  Then Yellow F-Line A,B,C,D↵
IfEnd↵

Goto 1↵
```

The basic structure of this program:

1. Setup various things
2. Place a Lbl (label) to 'go (back) to' at some point.
3. Define 5 variables.
4. Make seven If–Then-IfEnd statements based on the value of P.
5. Draw a line, the colour of which is determined by the value of P.
6. Go back (Goto) to Lbl 1

The process continues ad infinitum or until you break it or until the batteries give up. ☺

Note that Goto statements are used in this book to reduce the cognitive load for beginning programmers.

Exercise 1

In the program list you will be able to find programs named B~RANDL*, and RANDC*.

Execute each of them and see how they behave.
Predict how the code will be different to that seen opposite.

Explore the code of each and see if you were correct.

In this program we have chosen to define the colour of the lines using a different method than that used in A~PROG1.

- `Plot/Line-Color Magenta↵` , as seen in A~PROG1, is a 'global' function.
- `Cyan F-Line a,b,c,d↵` , as seen in B~RANDL3, is a 'local' function.

Using 'local' functions results in faster running programs, but takes a little more effort to enter.
All Sketch functions (e.g. `F-Line a,b,c,d`, `Circle x,y,r`, `Plot x,y`) can have a colour precede them in a line of code.

Note
Since this program **sets** Axes Off, etc., and these settings are global settings, when you use another application on the fxCG50 AU immediately after running this program you may have to reset the settings in that application.

3.1 About entering the code for the program named B~RANDL3.

If you are wondering how all this code is entered, do not worry, you will not have to do that for a while! You will be modifying pre-entered code for many activities in this book.

So you are aware, however, when starting from scratch, code is entered by:
a) finding a function/command in a menu and entering it as a "block" and
b) typing in numbers/letters from the keyboard.

`"PRESS EXE TO START"`
Is typed in, using the red "ALPHA" keys to enter text and quotes.

◢ is entered as a "block" and is found by pressing:
PRGM (SHIFT then VARS) then F5.

`Cls`, like many other commands, cannot be entered by typing, but is entered as a "block".
It can be found, and entered, by pressing:
Sketch (Shift F4) then Cls (F1).

`ViewWindow a,b,c,d,e,f` is entered as a block by pressing:
V-Window (Shift F3) then V-Win (F1).
The values of a, b, c, d, e & f are entered from the keyboard.

`Setup commands`
SET UP (Shift then MENU).

`Lbl-Goto`
PRGM (Shift then VARS) Then JUMP (F3).

`RanInt#(a,b)`
OPTN then F6 then PROB (F3) then RAND (F4) then Int (F2).

`If-Then-IfEnd`
PRGM (Shift then VARS) then COMMAND (F1) then choose.

`Plot/Line-Color colour`
SETUP (SHIFT then MENU) then F6, F6, F6, F6, F6, F3.

`colour`
FORMAT (SHIFT then 5) then 1 then choose your colour.

`F-Line a,b,c,d`
Sketch (SHIFT then F4) then F6 then LINE (F2) then F-Line (F2).

`Plot x,y`
Sketch (SHIFT then F4) then F6 then PLOT (F1) then Plot (F1).

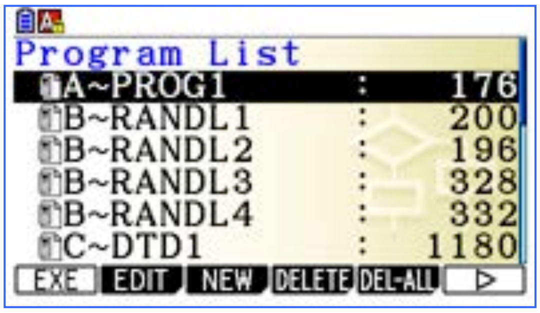
`Circle x,y,r`
Sketch (SHIFT then F4) then F6 then Circle (F3).

Alternatively, commands can be found in the catalogue:
CATALOG (SHIFT then 4).
The red letters can be used to jump to the correct alphabetic section of the catalogue.

If you are a seasoned programmer, it is possible to write the code in a text file on a computer and then load the text file into either the fx-CG50AU handheld or the fx-CG Manager PLUS software. This is discussed in more detail later in this document.

If you are a beginner, then it is time to borrow another person's code and modify it! ☺

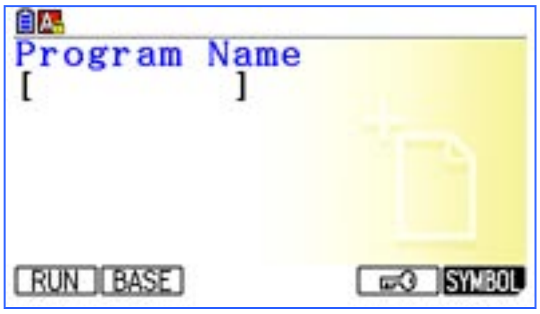
4. Borrowing and modifying code.



```
Program List
A~PROG1    :  176
B~RANDL1   :  200
B~RANDL2   :  196
B~RANDL3   :  328
B~RANDL4   :  332
C~DTD1     : 1180
```

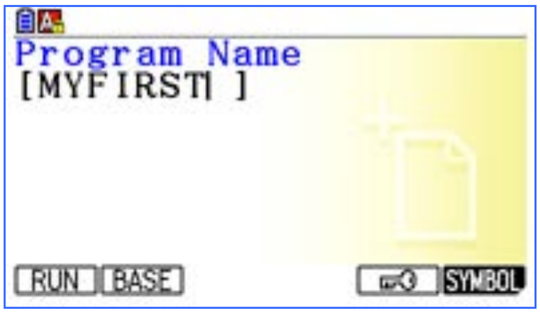
Launch the Program application.

Choose NEW (F3) to start making a new script/program.



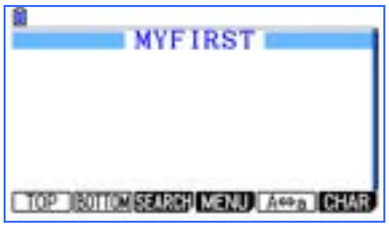
```
Program Name
[          ]
```

Note the A in the red box, with a yellow lock, top left of screen. All keys on the machine act as the red letters when this is active – it is auto active when you start a new program.



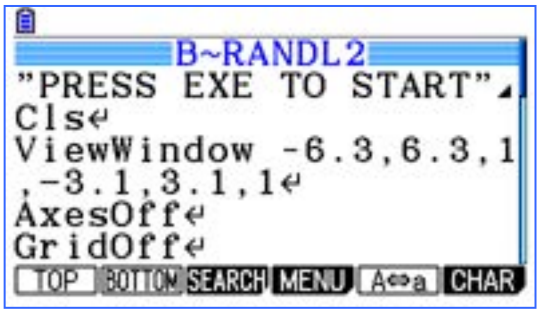
```
Program Name
[MYFIRST  ]
```

Type in the name MYFIRST and press EXE.



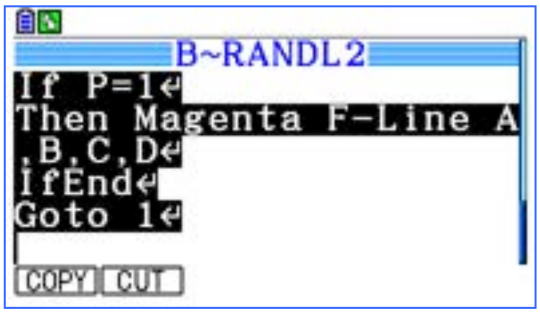
```
MYFIRST
```

You are now ready to borrow some code.



```
B~RANDL2
"PRESS EXE TO START"◢
Cls↵
ViewWindow -6.3,6.3,1
,-3.1,3.1,1↵
AxesOff↵
GridOff↵
```

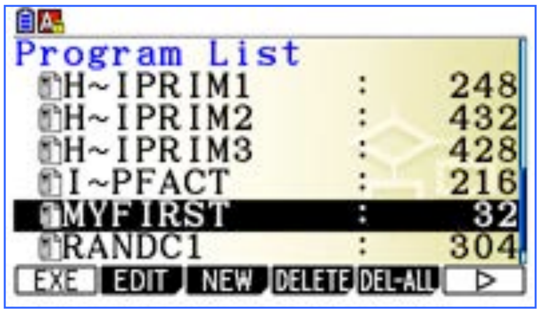
Press EXIT and locate B~RANDL2. Press EDIT (F2).



```
B~RANDL2
If P=1↵
Then Magenta F-Line A
,B,C,D↵
IfEnd↵
Goto 1↵
```

We want to copy all the lines of code in this program.
Press CLIP (SHIFT then 8) and then use the down cursor key to select all lines.

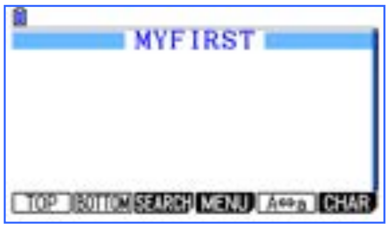
Now choose COPY (F1).



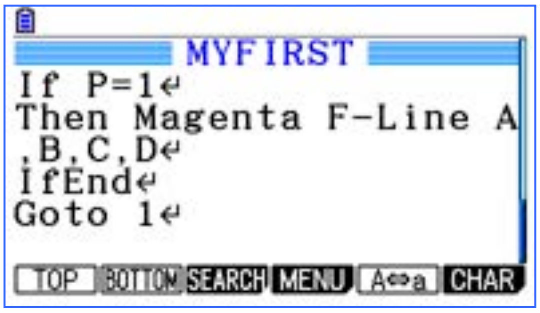
```
Program List
H~IPRIM1      :  248
H~IPRIM2      :  432
H~IPRIM3      :  428
I~PFACT       :  216
MYFIRST       :   32
RANDC1        :  304
```

Press EXIT to go back to the 'choose a program' (Program List) screen.

Highlight MYFIRST and press EDIT (F2).



```
MYFIRST
```



```
MYFIRST
If P=1↵
Then Magenta F-Line A
,B,C,D↵
IfEnd↵
Goto 1↵
```

Now use PASTE (SHIFT then 9) to paste the code.

You are now ready to modify this code! ☺

Oh, one last thing – if you want to change the colour of objects, you will need to know the following:

Regarding Text colour (and all Sketch objects) the line of code:
`"PRESS EXE TO START"◢`
Results in the message being displayed in black.

`Blue "PRESS EXE TO START"◢`
will be displayed in blue.

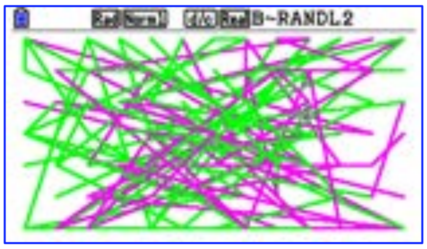
`Red "PRESS EXE TO START"◢`
will be displayed in red.

The colour command is inserted by using:

- FORMAT (SHIFT then 5),
- Color Command (1) and then choose the colour you want.

Exercise 2

The code you have copied into MYFIRST will execute the same way as B~RANDL2.

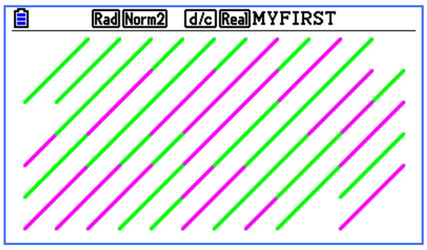


The start and end points of each line are not related (independent of each other).
As such the lines are "all over the place".

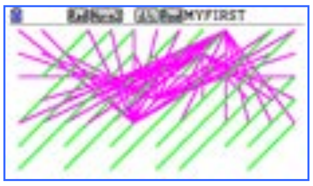
Your task is to change the start and end points, so they are related in some way.

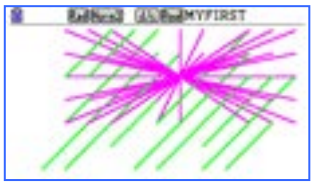
For example, not
`Magenta F-Line A,B,C,D`
but
`Magenta F-Line A,B,B,A`
or some other approach; you could try many different things, maybe
`Magenta F-Line A,B,2A,B`

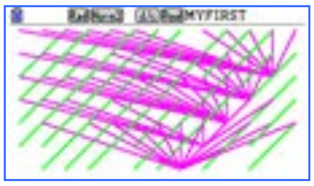
I tried various approaches and came up with some lovely art, shown below.
First this:



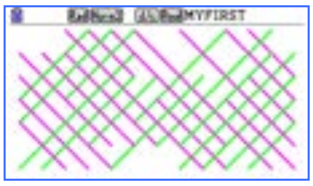
and then these:



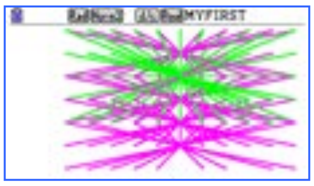


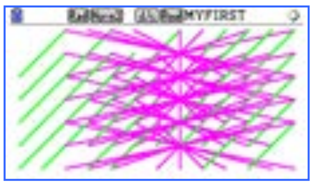


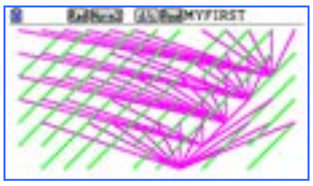
and then this:



which is what I was aiming to do first, but got all those other nice patterns while making errors.
Here are some more I made.







What can you create?

Exercise 3

Make a new program called SCWORD.

SCWORD is short for screen-word (or showing a word on screen).

Use
`F_Line a,b,c,d`
Sketch (SHIFT then F4) then F6 then LINE (F2) then F_Line (F2),
And other commands to create a program that displays EAT! on screen, made from coloured lines.

You may like to start by using paper and pencil to draw a cartesian plane so you know where to start and end lines.

Exercise 4

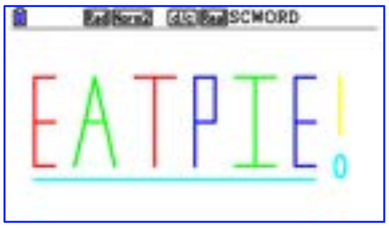
Repeat Exercise 3, but for a word of your choosing, complete with and exclamation mark.

Exercise 5

Repeat Exercise 4, but this time make the program display a flashing exclamation mark.

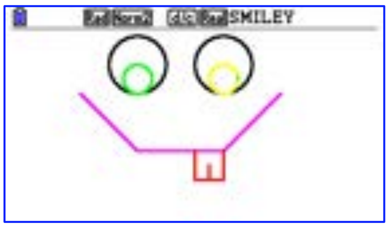
To do this draw the exclamation mark in a given colour and then re-draw it in a different colour and repeat this sequence over and over using:
`Lbl-Goto`
PRGM (Shift then VARS) Then JUMP (F3).
as seen in previous programs.

The "flashing" will be quite slow on a fx-CG50AU hand-held but fast if using the fx-CG Manager PLUS.



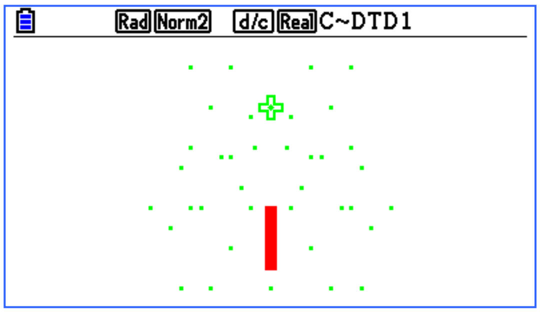
Once you have succeeded, add some other flashing elements to your creation.

You may want to graduate from words to a smiling face with a tongue hanging out – like some students have done! Below you can see the yellow eye-inner starting to change from yellow to green.



5. Dot-to-dot and Pause.

Have you ever done a dot-to-dot drawing?
Can you imagine making an e-version of dot-to-dot, but with a difference, given the machine will be doing most of the work. How do we ensure the element of surprise for the doer?



Run the program C~DTD1.

Note that the program *pauses* after drawing a red rectanlge and some dots.

What could the drawing be, if the dots were joined with straight lines in some predetermined order?

To make the program continue, press EXE.

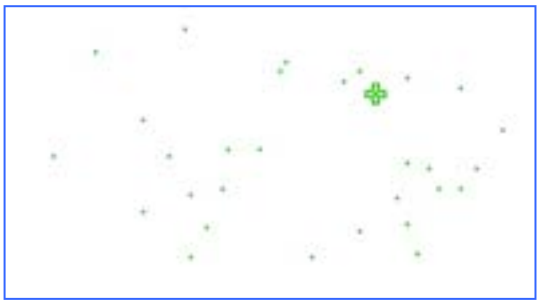
Ooooo – scary!

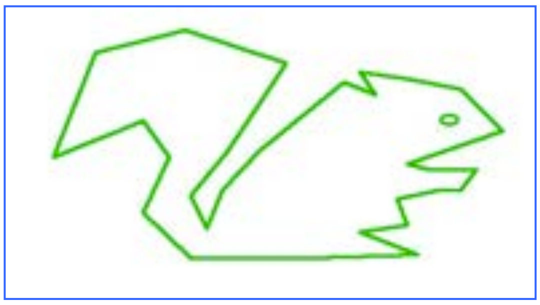
The complete program can be seen in Appendix 1.

The *pause* is actioned by the **display** command (◢), as seen in the lines of code below. Whatever the machine is commanded to draw before the ◢ is displayed on screen.

```
.
.
.
PlotOn -0.4,0.5↲
PlotOn -0.5,1.25↲
PlotOn 0,1.5◢
Plot/Line-Color Black↲
F-Line 0,1.5,0.5,1.25↲
F-Line 0.5,1.25,0.4,0.5↲
.
.
.
```

Students, having seen the above example, were keen to create their own dot-to-dot. Below you can see the dot part of a student's e-dot-to-dot, can you guess what it is?





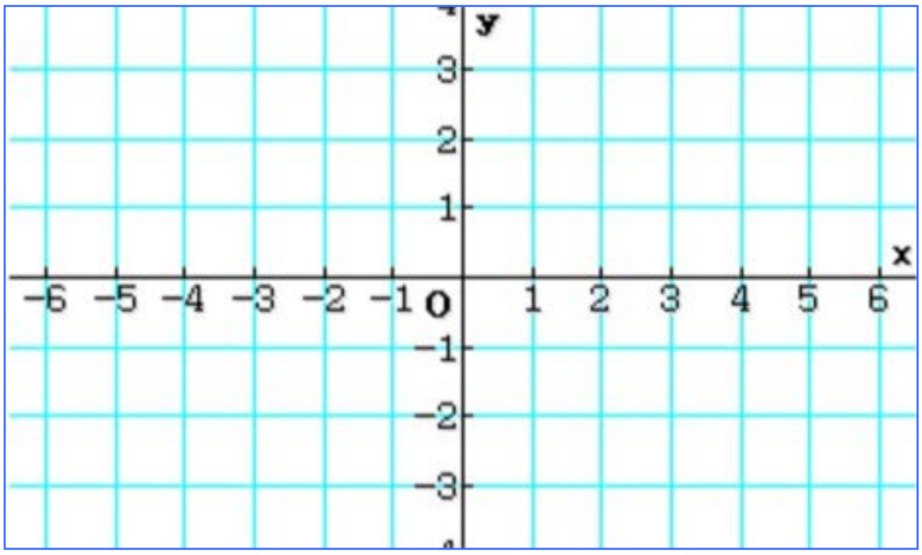
Oh bless, a cute wee squirrel!

I wonder why his eye is like that? It is hardly round.
The function `Circle x,y,r` was used to draw the eye.
Can you image what the student has done? Maybe something to do with
`ViewWindow xmin,xmax,scale,ymin,ymax,scale`

Exercise 6

Create your own e-dot-to-dot.

Start with a paper grid with axes draw on it, something like that seen below, and then draw your 'critter' and carefully record the coordinates that form it.



Then start by writing at least the basic structure of the program on paper, before you start to enter it into the machine.

Once it is working, share your e-dot-to-dot with your friends.

Note
The colour in the program C~DTD1, seen in Appendix 1, is handled with the global command, i.e.
`Plot/Line-Color Black`
Since fast execution of code is not critical in this case it is better (from a code entry point of view) to use a single line of code for 'colour definition'.

6. The While loop – adding on and on and on …

Consider following:

- $\frac{1}{4}$
- $\frac{1}{4}+\frac{1}{4\times4}$
- $\frac{1}{4}+\frac{1}{4\times4}+\frac{1}{4\times4\times4}$

Each of these are sequential steps in a growing string of numbers that are being added together. In each step we add on one number and we keep doing this forever. If the denominator of the next fraction always has one more $\times 4$ and the numerator is constantly 1, then the "forever" version can be written as follows:

$$\frac{1}{4}+\frac{1}{4\times4}+\frac{1}{4\times4\times4}+\frac{1}{4\times4\times4\times4}+\ldots\ldots+\frac{1}{4^n}$$

where n is a positive whole number.

Complete the following table for $1 \leq n \leq 8$, giving both fractional and decimal values.

n	1	2	3	4	5	6	7	8	. . .
sum (S)	$\frac{1}{4}$ 0.25	$\frac{5}{16}$ 0.3125							. . .

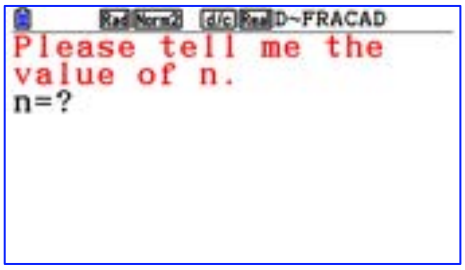
What do you notice?

Can you suggest what will happen to the value of the sum, S, as the value of n becomes larger and larger (as $n \to \infty$)?

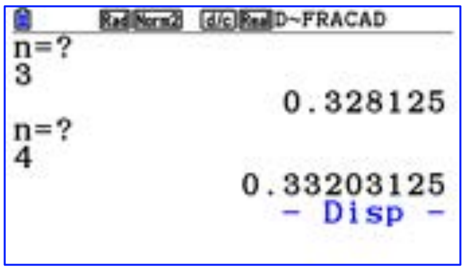
The program called D~FRACAD will add up a given number of steps.
You need to supply the number of steps (the value of n).
D~FRACAD will help you to be more confident about your suggestion from the previous page.

The program uses a While loop to add-on the next step.
The program is shown below.

```
Norm 2↵

Red "Please tell me the   value of n."↵
Lbl 1↵
"n="?→N↵

0→A↵
4→B↵
1→C↵

While C≤N↵
   A+(1/B)→A↵
   B*4→B↵
   C+1→C↵
WhileEnd↵

A◢
Goto 1↵
```



```
D~FRACAD
Please tell me the
value of n.
n=?
```



```
D~FRACAD
n=?
3
                 0.328125
n=?
4
               0.33203125
                   - Disp -
```

The calculator repeats the calculations in the While loop until C>N (while C≤N).

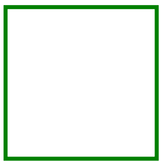
C is the variable that counts the number of times the calculations are done.

`While-WhileEnd` can be found by:
PRGM (SHIFT then VARS) then COMMAND (F1) then F6 then F6 then F1 or F2,
or in the CATALOG (SHIFT then 4).

`≤` can be found by:
CHAR (F6) when the code of a program is displayed

So what appears to happen to the value of the sum as the value of n becomes larger and larger (as $n \to \infty$)?

Can you "fraction-up" the square, shown below, in a way that would convince another person that as $n \to \infty$, $S \to \frac{1}{3}$?



Note 1
The empty lines in the program shown above are added to help you read the code. They are allowed in the fx-CG50AU but require a carriage return. The shading and indents, however, are not allowed. ☺

Note 2
The extra spaces between 'the' and 'value' (in the second line of code) are present because the programming application does not include a clever word wrapping process. The screen is 21 characters wide and therefore some thought is required to include the wrapping process in your code.

Exercise 7

Consider the sum:

$$S_n = \frac{1}{5} + \frac{1}{5\times5} + \frac{1}{5\times5\times5} + \ldots\ldots + \frac{1}{5^n}$$

Make a new program to compute the sum for chosen values of n.
I suggest you make a new program, then copy and paste the code from D~FRACAD into it and then modify! ☺

What happens to the value of S_n as $n \to \infty$?

Exercise 8

Consider the sum:

$$S_n = \frac{1}{6} + \frac{1}{6\times6} + \frac{1}{6\times6\times6} + \ldots\ldots + \frac{1}{6^n}$$

What happens to the value of S_n as $n \to \infty$?

Exercise 9

Consider the sum:

$$S_n = \frac{1}{7} + \frac{1}{7\times7} + \frac{1}{7\times7\times7} + \ldots\ldots + \frac{1}{7^n}$$

What happens to the value of S_n as $n \to \infty$?

Exercise 10

Consider the sum:

$$S_{a,n} = \frac{1}{a} + \frac{1}{a\times a} + \frac{1}{a\times a\times a} + \ldots\ldots + \frac{1}{a^n}$$

Can you suggest what happens to the value of $S_{a,n}$ as $n \to \infty$?

Can you prove that your suggestion is correct?

Exercise 11

Consider the sum:

$$S_n = 1 + \frac{2}{3} + \frac{4}{9} + \frac{8}{27} + \ldots\ldots + \frac{2^{n-1}}{3^{n-1}}$$

What happens to the value of S_n as $n \to \infty$?

7. If-Then-Else-IfEnd within a While loop. Can you guess my number?

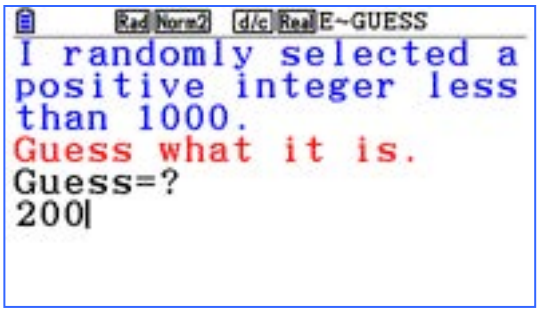
The program named E~GUESS, randomly selects a positive integer from the set {1,2,3, …, 1000} and prompts you to guess the integer.

When you enter your guess, the program determines if your guess was greater than or less than the positive integer it selected.

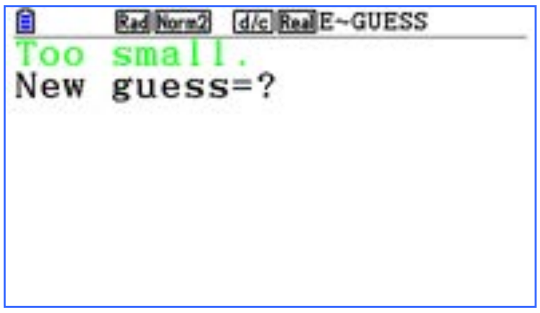
You are then prompted to guess again, and so on, until you enter a correct guess.

The program will then report how many guesses you made.

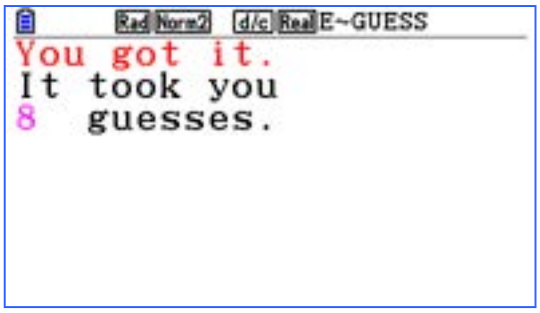
Play this game a few times. Play a few more times and see how good you can become.



```
I randomly selected a
positive integer less
than 1000.
Guess what it is.
Guess=?
200
```



```
Too small.
New guess=?
```



```
You got it.
It took you
8   guesses.
```

Exercise 12

Before turning the page to look at the code in E~GUESS, sketch out what you think the program is doing and what commands might be required.

The code below is the main part of E~GUESS.

```
0→N
RanInt#(1,1000)→S

ClrText
Blue "I randomly selected apositive integer lessthan 1000."
Red "Guess what it is."
"Guess="?→G

While G≠S
1+N→N

  If G<S
    Then ClrText
      Green "Too small."
  Else ClrText
      Red "Too large."
  IfEnd

WhileEnd

ClrText
Red "You got it."
"It took you"
"   guesses."
Magenta Locate 1,3,N
```

`RanInt#` can be found by:
OPTN then F6 then PROB (F3) then RAND (F4) then RAND# (F1),
or in the CATALOG (SHIFT then 4).

The If-Then-Else-IfEnd block repeatedly reports if the current guess is too small or too large, while G≠S.

The program pauses and waits for you guess. However, there are no pause/display commands (◢) in this script; the *input* command line `"Guess="?→G`, actions a pause.
`?` can be found by:
PRGM (SHIFT then VARS) then ? (F4),
or in the CATALOG (SHIFT then 4).

The line `Magenta Locate 1,3,N`, prints the current value of N at the screen location 1,3.
The 3 refers to the row number, from the top of the screen; the 1 refers to the position in a row, from the left-most position of a row.

`Locate` can be found by:
PRGM (SHIFT then VARS) then F6 then I/O (F4) then Locate (F1),
or in the CATALOG (SHIFT then 4).

Exercise 13

A bug exists in this code. If the user guesses correctly on their first guess, the program does not quite work correctly. Study the code and determine what happens. Squish the bug!

Note 1
Carriage returns (↵) are missing from the end of each line, above, to improve the readability of the code.

Note 2
The order of the last two lines of code may appear odd – it is done so that the way the screen message unfolds looks the best it can in this platform.

The complete form of E~GUESS is given below.
The additional lines, to those shown on the previous page, are shown in bold.

```
0→N
RanInt#(1,1000)→S

Lbl 1

ClrText
Blue "I randomly selected apositive integer lessthan 1000."
Red "Guess what it is."
"Guess="?→G

If Int G≠G
  Then Goto 1
IfEnd

While G≠S
1+N→N

  If G<S
    Then ClrText
      Green "Too small."
  Else ClrText
      Red "Too large."
  IfEnd

  Lbl 2
  "New guess="?→G
  If Int G≠G
    Then Goto 2
  IfEnd

WhileEnd

ClrText
Red "You got it."
"It took you"
"   guesses."
Magenta Locate 1,3,N
```

The additional lines act as a catch, to stop the user entering a guess that is not an integer.

The command `Int G` calculates the integer part of the current value G.
For example `Int 2.34 = 2`
`Int` can be found by:
OPTN then F6 then NUMERIC (F4) then Int (F2),
or in the CATALOG (SHIFT then 4).

Exercise 14

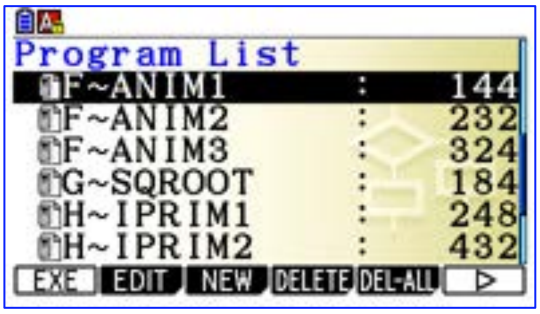
A very much simpler form of this guessing/checking process would be for the program to randomly choose the number 0 or 1 and then the player has to guess which number the program chose.

Make a program that performs this simpler guessing/checking process 5 times in a row and the program reports how many times the player guessed correctly.

Once you succeed, modify your program so the player can set the number of times the guessing/checking process occurs.

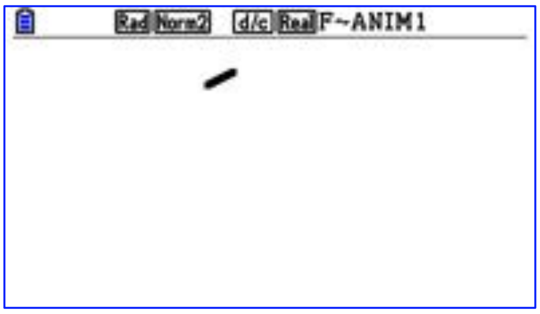
8. Animation, using a For-Next loop.

Run the program F_ANIM1.


Program List

```
F~ANIM1   : 144
F~ANIM2   : 232
F~ANIM3   : 324
G~SQROOT  : 184
H~IPRIM1  : 248
H~IPRIM2  : 432
```

While the image seen right does not show the action, it does display one 'frame' of the action.



The illusion of movement (animation) is created by the process of drawing an image, erasing that image and drawing another image nearby and then repeating this over and over.

Did you notice that the projectile changes length, as though it is elastic?

Here are the lines of code that produces the animation. Not too many at all.

```
"PRESS EXE TO START"◢

ViewWindow -4,4,1,-8,4,1↵
AxesOff↵
GridOff↵
CoordOff↵
LabelOff↵
S-L-Thick↵
Plot/Line-Color Black ↵

Lbl 1↵
Cls↵

For -3→A To 3 Step 0.5↵
  F-Line A,3-A^2,A+0.4,3-(A+0.4)^2↵
  Cls↵
Next↵

Goto 1↵
```

Reminder
Since this program **sets** Axes Off, etc., and these settings are global settings, when you use the fxCG50 AU immediately after running this program you may have to reset the settings in the application you are using.

The heart of this program are the following lines of code.

```
For -3→A To 3 Step 0.5
  F-Line A,3-A^2,A+0.4,3-(A+0.4)^2
  Cls
Next
```

The `F-Line` command is executed over and over again under the direction of the For-Next loop. The variable A controls both how many times the For-Next loop is repeated; it also controls the position of the start and end of the line drawn.

`For-Next` can be found by:
PRGM (SHIFT then VARS) then COMMAND (F1) then F6,
or in the CATALOG (SHIFT then 4).

Exercise 15

Change the value of 0.5 to 1 in the line

```
For -3→A To 3 Step 0.5
```

Describe the effect of this change. Experiment with other values to create the opposite effect.

Exercise 16

Time to spice this program up a little.

a) Modify the program so the projectile is a colour other than black.
b) Modify the program so the projectile is a different colour on the way up to the way down.
c) Modify the program so the two projectiles are launched, one from left to right and one from right to left.

Exercise 17

Modify the program so the projectile's path is something other than a parabola.
For example, change 3-A^2 into 3-A.
Can you predict what change that will make to the projectile's path?
Experiment with other changes that give different paths.

Exercise 18

Run the program **F_ANIM2**.
Does it provide the illusion of 3-dimensional motion to you; a falling-spinning stick?

The code for **F_ANIM2** can be found in Appendix 2.

Also included in the program set is a program called **F_ANIM3**, which results in the same animation as **F_ANIM2**, but it handles colouring the lines in the alternate manner.
Do you notice a difference in the speed of rotation?

9. Square roots, using a For-Next loop.

Have you ever wondered how a calculator figures out a decimal approximation for the square root of a number – e.g. $\sqrt{24}$?

One way is for it to carry out a recursive process that can be described as follows:

- take a guess at the answer,
- do a calculation using the guess,
- the calculation gives a result that is closer to the actual value,
- use this result to do the calculation again and then
- keep repeating this over and over until the newest result is close (enough) to the actual value.

If we wish to calculate an approximation for $\sqrt{S}$, the guess is x_n and the first/next result is x_{n+1} then the calculation and process can be described as

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right)$$

This process is the result of using Newton's Method to solve $x^2 - S = 0$.

You can read about it at https://en.wikipedia.org/wiki/Methods_of_computing_square_roots.

Before we use a program to do this, we will first do some calculations, so we gain an appreciation of the process.

Exercise 19

Let S =17 and let the guess, x_0, for the value of $\sqrt{17}$ be half of 17 (8.5).
One step in the recursive process is called an iteration.
Do 5 iterations.
Iteration one and two are given below.
Finish off the process.

Iteration 1:

$$x_1 = \frac{1}{2}\left(x_0 + \frac{S}{x_0}\right)$$

$$x_1 = \frac{1}{2}\left(8.5 + \frac{17}{8.5}\right) = 5.25$$

Iteration 2:

$$x_2 = \frac{1}{2}\left(x_1 + \frac{S}{x_1}\right)$$

$$x_2 = \frac{1}{2}\left(5.25 + \frac{17}{5.25}\right) = 4.244047619$$

Iteration 3:

$$x_3 = \frac{1}{2}\left(x_2 + \frac{S}{x_2}\right)$$

Compare your value for x_5 with that of other people.

4.123106 is the value of $\sqrt{17}$, correct to six decimal places. How close was x_5 to 4.123106?

The recursive (looping) nature of this process lends itself to computer automation.
A For-Next loop can be employed.

The program called G~SQROOT automates this process.

In the previous exercise you were told to make the guess one-half of the value of S.
We use the same guess in this code.

The main part of the code is shown below.

```
ClrText↵
Norm 2↵

Blue "You require the Sqrt of"↵
?→S↵

S/2→X↵

ClrText↵
Blue "How many iterations"↵
?→I↵

For 1→L To I Step 1↵
   (1/2)*(X+(S/X))→X↵
   X◢
Next↵
```

The heart of this program is the following lines of code.

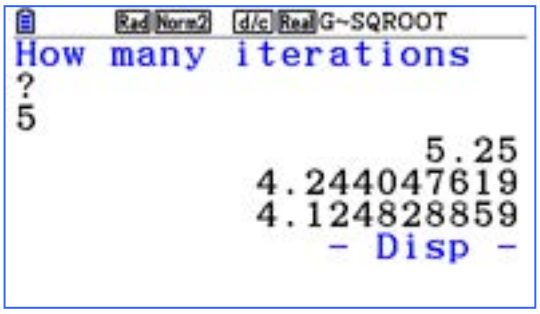
```
For 1→L To I Step 1
   (1/2)*(X+(S/X))→X
   X◢
Next
```

The calculation `(1/2)*(X+(S/X))→X`, is executed over and over again under the direction of the For-Next loop.
The variable L controls how many times the For-Next loop is repeated.

`For-Next` can be found by:
PRGM (SHIFT then VARS) then COMMAND (F1) then F6,
or in the CATALOG (SHIFT then 4).

Run G~SQROOT and calculate $\sqrt{17}$ with 5 iterations.

Check the output against your calculations made the previous exercise.
A partial output is shown opposite.



```
Rad Norm2 d/c Real G~SQROOT
How many iterations
?
5
                  5.25
           4.244047619
           4.124828859
               - Disp -
```

The complete G~SQROOT program is shown below.

Note the bold block, which was not shown on the previous page.

```
ClrText↵
Norm 2↵

Blue "You require the Sqrt of"↵
?→S↵

S/2→X↵

Lbl 1↵
ClrText↵
Blue "How many iterations"↵
?→I↵

If I>40↵
  Then Cls↵
    "Please choose fewer."◢
    Goto 1↵
IfEnd↵

For 1→L To I Step 1↵
  (1/2)*(X+(S/X))→X↵
  X◢
Next↵
```

The If-IfEnd block seen left, is a catch for an excessive value of I (the number of iterations).

Exercise 20

Suppose you want to find the $\sqrt{2\ 000\ 000\ 000}$.
Use G~SQROOT to determine how many iterations are required to find $\sqrt{2\ 000\ 000\ 000}$ such that the 4th decimal place is correct.

Exercise 21

Experiment with numbers of the form $2\ \times 10^n$ to see whether or not a relationship exists between the number of iterations and n, for a give degree of accuracy is required.

10. If-Then-IfEnd inside a For-Next; is that number prime?

Is 65 701 a prime number?

One way to figure out whether or not 65 701 is prime is to divide it by 2, 3, 4, 5, ….. , int($\sqrt{65\,701}$).
That is quite a bit of work and something a computer can do.

Advanced calculators and some mathematical software are equipped with a function called something like "isPrime", which simply states prime or not. The user, however, rarely gets to see how the function actually works, it is just black-box magic!

H~IPRIM1 is a program, shown below, that actions one algorithm that determines whether or not a positive integer, N, is prime.

The algorithm divides N by 2, then 3, then 4, then 5, then ….. and stops after reaching the integer value of $\sqrt{65\,701}$.

```
ClrText↵
Lbl 1↵
Blue "Which integer do you want to test for     primality"↵
?→N↵

If Int N≠N↵
   Then ClrText↵
   "That is not an integer!"↵
   Goto 1↵
IfEnd↵

For 2→I To Int (√N) Step 1↵

   If Int (N/I)=N/I↵
      Then ClrText↵
      " "
      Locate 1,1,N↵
      "is not prime."↵
      Goto 2↵
   IfEnd↵

Next↵

ClrText↵
" "
Locate 1,1,N↵
"is prime."↵
Lbl 2↵
Stop↵
```

Note the block in red print is a catch for non-integer entries.

Also note that the If-Then-IfEnd block in green print lies inside the For-Next loop.
Thus the If-Then-IfEnd block is executed every time the For-Next loop loops.
In this program, the For-Next loop does not necessarily loop for the maximum possible number of times, as set by the value of N, because of the presence of the Goto jump.
The `Goto 2` line, if actioned, will short cut the process, as we only need to find one number that divides N with zero remainder.

Exercise 22

Run H~IPRIM1 and try it out for numbers, **both prime and not prime** under 10,000.

Some primes under 10,000 are:
3659
6763
8429
9931

Be sure to try some composite numbers to check the program has no bugs.

Write down the time taken by the longest computation.

Exercise 23

Now try some larger primes but under 100,000. Here are three:

- 30011
- 68909
- 97579

Write down the time taken by the longest computation.

Exercise 24

What about 1246537?
Is it prime? How long did it take to determine whether or not it was?

Exercise 25

Test each of the following numbers for primness and record the time taken (t) by the program to determine whether or not the number was prime.

- 7
- 73
- 739
- 7393
- 73939
- 739391
- 7393913
- 73939133

Let l be the logarithm of the number being tested.
Draw a graph of l vs t.

Exercise 26

Run H~IPRIM1 again and this time test a negative number for primness, e.g. – 29 .
Describe what happened and why it happened.
Modify the program to include a catch for entering negative numbers.
You might find the absolute value function (Abs) a useful tool.

`Abs` can be found by:
OPTN then F6 then NUMERIC (F4) then Abs (F1),
or in the CATALOG (SHIFT then 4).

Exercise 27

H~IPRIM1, does not contain a fast algorithm.
If it determined that 2 did not divide N, then there is no point in dividing by all of the other even numbers, as it currently does.
Modify this program so that it avoids doing all those unnecessary divisions by even numbers.
Does this modification result in a speed improvement?

Exercise 28

Consider the program H~IPRIM2, that is shown below.
It employs a different algorithm than that used in H~IPRIM1.
Study the code and describe the different algorithm.
Determine whether or not the algorithm used is faster than the one used in H~IPRIM1 by using the numbers from Exercise 24.

```
ClrText
Lbl 1
Blue "Which integer do you want to test for     primality?"
?->N

If Int (N)<>N
Then ClrText
Red "That is not an       integer!"
Goto 1
IfEnd

For 2->I To 3 Step 1
   If Int (N/I)=N/I
     Then ClrText
     " "
     Locate 1,1,N
     "is not prime."
     Goto 2
   IfEnd
Next

For 1->I To Int ((Sqrt(N)+1)/6) Step 1
   If Int (N/(6I-1))=N/(6I-1)
     Then ClrText
     " "
     Locate 1,1,N
     "is not prime."
     Goto 2
   IfEnd
Next

For 1->I To Int ((Sqrt(N)+1)/6) Step 1
   If Int (N/(6I+1))=N/(6I+1)
     Then ClrText
     " "
     Locate 1,1,N
     "is not prime."
     Goto 2
   IfEnd
Next

ClrText
" "
Locate 1,1,N
"is prime."
Lbl 2
Stop
```

Exercise 29

Consider the program H~IPRIM3, that is shown below. It takes a slightly different approach to H~IPRIM2. Study the code and explain the different approach.
Try it out. Is it faster than H~IPRIM2? Document your findings.

```
ClrText
Lbl 1
Blue "Which integer do you want to test for     primality?"
?->N

If Int (N)<>N
  Then ClrText
  Red "That is not an
  integer!"
  Goto 1
IfEnd

For 2->I To 3 Step 1

  If Int (N/I)=N/I
    Then ClrText
    " "
    Locate 1,1,N
    "is not prime."
    Goto 2
  IfEnd

Next

1->I

While (6I-1)<=SqrtN

  If Int (N/(6I-1))=N/(6I-1)
    Then ClrText
    " "
    Locate 1,1,N
    "is not prime."
    Goto 2
  IfEnd

I+1->I
WhileEnd

1->I

While (6I+1)<=SqrtN

  If Int (N/(6I+1))=N/(6I+1)
    Then ClrText
    " "
    Locate 1,1,N
    "is not prime."
    Goto 2
  IfEnd

I+1->I
WhileEnd

ClrText
" "
Locate 1,1,N
"is prime."
Lbl 2
Stop
```

Use H~IPRIM3 to test these two monsters:

- 591558727
- 5915587277.

Time accurately.

11. Whiles inside Ifs, (Whiles inside Ifs) inside Whiles – finding prime factors

The IPRIM* set of programs determines whether or not a positive integer is prime.
As such the algorithm need to only check up to the square root of N.
But what if we wanted to determine the prime factors of N, how could this be done, particularly if N was a large number, either prime or composite ?

The program I~PFACT determines the prime factors of positive integers.

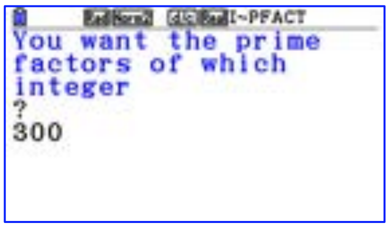
Exercise 30

Before looking at the code in I~PFACT (next page), figure out how you would determine, without a program, all of the prime factors of 300.

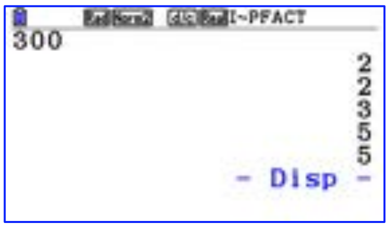
Write out a clear set of steps that a young person, who only knows the basics of number, could understand.

Exercise 31

Run the program I~PFACT.
Start with 'sanity' check with numbers – like 25 or 50 – to be sure you are confident the program is working. Then try 300.



```
You want the prime
factors of which
integer
?
300
```



```
300
                  2
                  2
                  3
                  5
                  5
            - Disp -
```

Finally use I~PFACT to find the prime factorisation of each of the following numbers.

- 8
- 74
- 740
- 2310
- 7394
- 73941
- 739413
- 7394133
- 73939133

Exercise 32

Study the code below, from the program named I~PFACT.
Write down an explanation of how it works.
`Frac (X)` is a command that determines the fraction part of X.
For example, `Frac (12.8)` = 0.8

`Frac (X)` can be found by:
OPTN then F6 then NUMERIC (F4) then Frac (F3),
or in the CATALOG (SHIFT then 4).

```
ClrText
Blue "You want the prime   factors of which     integer"
?->N
2->D

If Frac (N/D)=0
  Then

  While Frac (N/D)=0
    D◢
    N/D->N
  WhileEnd

  3->D
  Else 3->D
IfEnd

While D<=N

  If Frac (N/D)=0
    Then

    While Frac (N/D)=0
      D◢
      N/D->N
    WhileEnd

    D+2->D
    Else D+2->D
  IfEnd

WhileEnd

"DONE"
```

Exercise 33

If you found I~PFACT to be slow, especially as the input integers became larger, then you have a challenge. Develop a faster algorithm; or google one and code it up.

12. Writing code in a text file and importing it to the fx-CG50AU

Entering code directly into a handheld calculator can be surprisingly easy if you are adept at finding all of the commands. See Appendix 4.

If you are a confident coder and have a good eye and high regard for correct syntax, it is possible to type the code on a computer in the form of a text file (.txt), and then transfer the text file onto the fx-CG50AU.
The .txt file will be converted to the correct calculator format (.g3m) as part of the importation process.

The following is an example of what a text file looks like. The code is from I~PFACT.

```
ClrText
Blue "You want the prime   factors of which     integer"
?->N
2->D
If Frac (N/D)=0
Then
While Frac (N/D)=0
DDispsN/D->N
WhileEnd
3->D
Else 3->D
IfEnd
While D<=N
If Frac (N/D)=0
Then
While Frac (N/D)=0
DDispsN/D->N
WhileEnd
D+2->D
Else D+2->D
IfEnd
WhileEnd
"DONE"
```

Full details about this process can be found at:
http://support.casio.com/storage/en/manual/pdf/EN/004/fx-CG50_Soft_v320_EN.pdf
Page 8-7 to 8-9 (p302 to p304) &
Page 8-60 (p355) (for special characters).

Appendix 1 – C~DTD1

```
"PRESS EXE TO START"◢
Cls
ViewWindow (-)6.3,6.3,1,-3.1,3.1,1
AxesOff
GridOff
CoordOff
LabelOff
S-L-Normal
Plot/Line-Color Red
F-Line (-)0.1,-1,(-)0.1,(-)2.5
F-Line 0,-1,0,(-)2.5
F-Line 0.1,-1,0.1,(-)2.5
Plot/Line-Color Green
Plot 0,1.5
Plot 0.5,1.25
Plot 0.4,0.5
Plot 1,0.25
Plot 1,2.5
Plot 1.5,1.5
Plot 1.25,0.25
Plot 2,0.5
Plot 2,2.5
Plot 2.25,0
Plot 0.75,(-)0.5
Plot 3,(-)1
Plot 2.25,(-)3
Plot 2.5,(-)1.5
Plot 2,(-)1
Plot 1.5,(-)3
Plot 1.75,(-)1
Plot 0.5,(-)1
Plot 1,(-)2
Plot 0,(-)3
Plot (-)1,(-)2
Plot (-)0.5,(-)1
Plot (-)1.75,(-)1
Plot (-)1.5,(-)3
Plot (-)2,(-)1
Plot (-)2.5,(-)1.5
Plot (-)2.25,(-)3
Plot (-)3,(-)1
Plot (-)0.75,(-)0.5
Plot (-)2.25,0
Plot (-)2,2.5
Plot (-)2,0.5
Plot (-)1.25,0.25
Plot (-)1.5,1.5
Plot (-)1,2.5
Plot (-)1,0.25
Plot (-)0.4,0.5
Plot (-)0.5,1.25
Plot 0,1.5◢
Plot/Line-Color Black
F-Line 0,1.5,0.5,1.25
F-Line 0.5,1.25,0.4,0.5
F-Line 0.4,0.5,1,0.25
F-Line 1,0.25,1,2.5
F-Line 1,2.5,1.5,1.5
```

```
F-Line 1.5,1.5,1.25,0.25
F-Line 1.25,0.25,2,0.5
F-Line 2,0.5,2,2.5
F-Line 2,2.5,2.25,0
F-Line 2.25,0,0.75,(-)0.5
F-Line 0.75,(-)0.5,3,(-)1
F-Line 3,(-)1,2.25,(-)3
F-Line 2.25,(-)3,2.5,(-)1.5
F-Line 2.5,(-)1.5,2,(-)1
F-Line 2,(-)1,1.5,(-)3
F-Line 1.5,(-)3,1.75,(-)1
F-Line 1.75,(-)1,0.5,(-)1
F-Line 0.5,(-)1,1,(-)2
F-Line 1,(-)2,0,(-)3
F-Line 0,(-)3,-1,-2
F-Line (-)1,(-)2,(-)0.5,(-)1
F-Line (-)0.5,(-)1,(-)1.75,(-)1
F-Line (-)1.75,(-)1,(-)1.5,(-)3
F-Line -1.5,(-)3,(-)2,(-)1
F-Line (-)2,(-)1,(-)2.5,(-)1.5
F-Line (-)2.5,(-)1.5,(-)2.25,(-)3
F-Line (-)2.25,(-)3,(-)3,(-)1
F-Line (-)3,(-)1,(-)0.75,(-)0.5
F-Line (-)0.75,(-)0.5,(-)2.25,0
F-Line (-)2.25,0,(-)2,2.5
F-Line (-)2,2.5,(-)2,0.5
F-Line (-)2,0.5,(-)1.25,0.25
F-Line (-)1.25,0.25,(-)1.5,1.5
F-Line (-)1.5,1.5,(-)1,2.5
F-Line (-)1,2.5,(-)1,0.25
F-Line (-)1,0.25,(-)0.4,0.5
F-Line (-)0.4,0.5,(-)0.5,1.25
F-Line (-)0.5,1.25,0,1.5
Plot/Line-Color Black
Plot (-)0.25,1.1
Plot 0.25,1.1
Plot 10,10
Stop
```

Note

A carriage return (↵) is missing from the end of each line in this code, but the “display” command (◢) is included.

Appendix 2 – F~ANIM2

```
"PRESS EXE TO START"◢
ViewWindow -4,4,1,-12,1,1
AxesOff
GridOff
CoordOff
Rad
LabelOff
S-L-Thick

Lbl 1
Cls

For 0->A To 7 Step 0.2

   If A<1.7
     Then Plot/Line-Color Blue
   IfEnd

   If (A>=1.7 And A<4.7)
     Then Plot/Line-Color Red
   IfEnd

   If (A>=4.7 And A<6)
     Then Plot/Line-Color Blue
   IfEnd

   F-Line 1.0cos A,0.7*sin (A)-2A,1.0cos (A+3.14),0.7*sin (A+3.14)-2A
   Cls

Next

Goto 1
```

Note 1
A carriage return (↵) is missing from the end of each line in this code, but the "display" command (◢) is included.

Note 2
The numbers within the For-Next loop are values of parameters that control the 'sticks' behaviour.

Appendix 3 – F~ANIM3

```
"PRESS EXE TO START"◢
ViewWindow -4,4,1,-12,1,1
AxesOff
GridOff
CoordOff
Rad
LabelOff
S-L-Thick

Lbl 1
Cls

For 0->A To 7 Step 0.2

  If A<1.7
    Then Blue F-Line 1.0cos A,0.7*sin (A)-2A,1.0cos (A+3.14),0.7*sin
    (A+3.14)-2A
  IfEnd

  If (A>=1.7 And A<4.7)
    Then Red F-Line 1.0cos A,0.7*sin (A)-2A,1.0cos (A+3.14),0.7*sin
    (A+3.14)-2A
  IfEnd

  If (A>=4.7 And A<6)
    Then Blue F-Line 1.0cos A,0.7*sin (A)-2A,1.0cos (A+3.14),0.7*sin
    (A+3.14)-2A
  IfEnd

  Cls

Next

Goto 1
```

Note 1
A carriage return (↵) is missing from the end of each line in this code, but the "display" command (◢) is included.

Note 2
The numbers within the For-Next loop are values of parameters that control the 'sticks' behaviour.

Appendix 4 – Functions and commands, where to find them

The functions and commands used in the programs in this book can be entered by either:

- locating them in the CATALOG (SHIFT then 4) or
- by locating them in a menu, as outlined below.

A function is a process that performs a task for which one or more arguments are required. E.g. function for drawing a circle is `Circle x,y,r` where (x , y)is the centre of a circle with radius r.

Brackets of the form `()` are required to be entered in the code, e.g. `RanInt#(a,b)`

A command is a direction that performs a task that requires no arguments. E.g. The clear screen command, `Cls`

`? (Input command)`
PRGM (SHIFT then VARS) then ? (F4).

`≤`
CHAR (F6) (When the script of a program is visible, many special characters can be found here, e.g. ≠.)

`Abs X`
OPTN then F6 then NUMERIC (F4) then Abs (F1),

`Cls`
Sketch (Shift F4) then Cls (F1).

`colour`
FORMAT (SHIFT then 5) then Color Command (X)

> `e.g. Blue`
> FORMAT (SHIFT then 5),
> Color Command (1) then choose colour.

`Circle x,y,r`
Sketch (SHIFT then F4) then F6 then Circle (F3).

◢ Display character
PRGM (SHIFT then VARS) then F5.

`F-Line a,b,c,d`
Sketch (SHIFT then F4) then F6 then LINE (F2) then F-Line (F2).

`For-Next`
PRGM (SHIFT then VARS) then COMMAND (F1) then F6.

`Frac X`
OPTN then F6 then NUMERIC (F4) then Frac (F3),

`Lbl-Goto`
PRGM (Shift then VARS) Then JUMP (F3).

`If-Then-IfEnd`
PRGM (Shift then VARS) then COMMAND (F1) then choose.

`?` Input command
PRGM (SHIFT then VARS) then ? (F4),

`Int X`
OPTN then F6 then NUMERIC (F4) then Int (F2).

`Lbl-Goto`
PRGM (Shift then VARS) Then JUMP (F3).

`≤`
CHAR (F6) (When the script of a program is visible, many special characters can be found here)

`Locate c,r,X`
PRGM (SHIFT then VARS) then F6 then I/O (F4) then Locate (F1),

`Norm 2`
SETUP (SHIFT then MENU) then F6 the DISPLAY (F1) then Norm (F3) then 2.

`Plot x,y`
Sketch (SHIFT then F4) then F6 then PLOT (F1) then Plot (F1).

`Plot/Line-Color colour`
SETUP (SHIFT then MENU) then F6, F6, F6, F6, F6, F3.

`RanInt#(a,b)`
OPTN then F6 then PROB (F3) then RAND (F4) then Int (F2).

`Setup commands`
SET UP (Shift then MENU).
  `e.g. AxesOff`

`S-L-Normal`
SET UP (Shift then MENU) the F6 then SKT/LIN (F2) then F1.

`Stop`
PRGM (SHIFT then VARS) then CONTROL (F2) then Stop (F4)

`ViewWindow xmin,xmax,scale,ymin,ymax,scale`
V-WIN (Shift F3) then V-Win (F1).

`While-WhileEnd`
PRGM (SHIFT then VARS) then COMMAND (F1) then F6 then F6 then F1 or F2.

For further information see Chapter 8 of
http://support.casio.com/storage/en/manual/pdf/EN/004/fx-CG50_Soft_v320_EN.pdf

Projects

Each of the following projects can be approached by creating a program that will do the repetitive parts of the work required.

Project 1 – Find the number

Find the smallest positive integer, n, for which $\sqrt{n} - \sqrt{n-1} < 0.01$.

Project 2 - DS

231 is a three-digit number.
The digit sum of 231 is 6.
How many three-digit numbers have a digit sum of 6?

How many three-digit numbers have a digit sum of 1?
How many three-digit numbers have a digit sum of 2?
How many three-digit numbers have a digit sum of 3?
…

You get the idea. Observe, wonder and make a conjecture. Answer the question, "Why?".

Project 3 - Sqtrinums

36 is both a square number and a triangle number. It is the smallest such number. Let such numbers be called sqtrinums.
1225 is the next smallest sqtrinum. Find the next four lowest sqtrinums.

Project 4 - Darts

Three darts are thrown at a rectangular board that is divided into three sections. If a dart lands in section 1 the player earns 8 points; 10 points are earned for section 2 and 12 points for section 3.

Assuming:

1) a shot is one throw of each dart
2) each dart lands in one of the three sections
3) the score for a shot is the sum of the points earned,

how many different scores are possible and what are they?

fx-CG50AU

For further support such as:

- "HOW TO" videos
- Free lesson ideas, with video support
- How to show the fx-CG50AU on the big screen
- The Prime Schools PLUS program
- PC fonts for worksheet production

visit

www.casioeducation.com.au