27 Error handling

So far you have seen that when the computer finds an error, it halts execution of the program and prints a message on the screen. Some errors are generated by incorrect programming and these are the ones that you have to correct. But what about errors which occur during the execution of a good program, because either the data is wrong or the user inputs something that the computer cannot handle?

Look at the following program:

```
10 REPEAT
20 INPUT "NUMBER ", N
30 L=LOG(N)
40 PRINT"LOG OF ";N;" IS ";L
50 UNTIL FALSE
```

This is simple enough. It takes a number from the keyboard, and gives you the log of that number. But, if you type a negative number, the program comes to a halt with:

```
Log range at line 30
```

The same thing happens if you type 0, or a character such as W, or a word such as TWELVE.

It is easy to trap such an error, and to print a message to tell the user what he or she has done wrong. Every error has an error number, which is stored in a variable called ERR. You will find a list of these errors in Appendix B. The number for the log error is 22, so we can alter the program as follows:

```
5 ON ERROR GOSUB 70
10 REPEAT
20 INPUT"NUMBER ",N
30 L=LOG(N)
40 PRINT"LOG OF ";N;" IS ";L
50 UNTIL FALSE
100 IF ERR=22 THEN PRINT"MUST BE A POSIT
IVE NUMBER > 0"
110 RETURN
```

Now, if an illegal input is made, the program simply prints a message telling you what you have done wrong, and carries on running. The trouble is, it carries on running, and running, and running. The ESCAPE key has no effect. This is because ESCAPE is treated as an error; it even has its own error code: 17. You'll have to press BREAK, then OLD RETURN to get the program back.

To overcome this, RETURN can be incorporated into line 70:

```
100 IF ERR = 22 THEN PRINT"MUST BE A POSIT
IVE NUMBER > 0":RETURN
```

Now, the ESCAPE key works, but no message is printed to say what happened, or what line. Don't forget to delete 110.

To print the message, the instruction REPORT is used.

```
110 REPORT
```

will print the message Escape on the screen.

The line number at which errors occur is stored in a variable called ERL. If you add a line 120:

```
120 PRINT" at line ";ERL
```

then the correct message will be printed up on the screen.

The above example is fairly simple, because there is only the one error which can occur. If you write a program that other people will use, you will have to think of all the possible errors that may occur, and trap them accordingly. In the example, the instruction used was

```
ON ERROR GOSUB
```

Other useful error trapping instructions are:

```
ON ERROR GOTO
ON ERROR PRINT
ON ERROR PROC
```

If you do use GOTO, you cannot 'GOTO' back again into a FOR . . . NEXT loop, a REPEAT . . . UNTIL loop, or a function. See Appendix B for a list of errors and their codes.