```
10 REM A PROGRAM TO LOCATE A NUMBER IN AN
   ARRAY
20 DIM SCORES(20)
30 FOR Z = 1 TO 20
40 READ SCORES(Z)
50 NEXT Z
60 DATA 0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,9,11,12
70 LET L = 20
80 LET BTM = 1
90 LET TP = L
100 INPUT"INPUT SCORE ";N
110 FOR Z = 0 TO 1 STEP 0
120 LET L = TP — BTM
130 LET MD = BTM + INT(L/2)
140 IF N = SCORES(MD) THEN LET Z = X
150 IF N > SCORES(MD) THEN LET BTM = MD
160 IF N < SCORES(MD) THEN LET TP = MD
170 NEXT Z
180 PRINT "THE SCORE WAS IN ELEMENT NO.
   ";MD
190 END
```

Again, note that X will need to be initialised according to your machine's requirements (see Basic Flavours).

If the data held in a file or array is fairly regular, as in the case of a telephone directory, where names are distributed reasonably evenly across the alphabet, then thebinary search is an efficient way of finding a particular entry. However, it is by no means the most efficient, and there are alternative algorithms that can find the data using fewer iterations. One such is the technique of 'hashing', where the program makes an educated guess at the approximate location of the entry, refining the guess until it is found. Such methods, however, are beyond the scope of this course, and the binary search method is sufficient for our needs.

Exercises

If you run this program, you will see that it works provided you enter a score that exists in the array. If you enter a score such as 3, which is not in the array, the program fails to terminate and no error message appears. If you type in 12, which exists in the array, the program fails to locate it. The program also assumes that every number in the sorted array will be different but, as you can see from the data statement, several numbers occur more than once. The program neither detects this nor reports all the locations where the number occurs.

Your task is to:

1. Analyse the program and find out why it cannot locate a score of 12
2. Modify one line of the program to rectify this defect
3. Establish why the program is unable to handle numbers that do not exist in the string and devise a strategy to overcome this defect.

On page 235 of The Home Computer Course we featured a number of revision exercises to help you assess your progress in the Basic Programming course. See page 280 for the solutions.

Basic Flavours

SPECTRUM

The following is the Spectrum listing for the first BASIC program fragment:

```
100 DIM F$(6,4)
110 LET POSITION = 0
120 LET F$(1) = "MIKE"
130 LET F$(2) = "KATE"
140 LET F$(4) = "MARY"
1000 FOR L = 0 TO 1 STEP 0
1010 LET POSITION = POSITION + 1
1020 IF CODE (F$(POSITION)) = 32 THEN
   LET L = 2
1030 NEXT L
1040 PRINT "NO. OF 1st FREE ELEMENT
   IS ";POSITION
1050 STOP
```

Notice that lines 100 to 140, as well as line 1040, transform the program fragment (lines 1000 to 1030) into a working demonstration program. The values and format of these extra lines can be changed to investigate the working of the program fragment.

The second program for the Spectrum is:

```
10 REM A PROGRAM TO LOCATE A
   NUMBER IN AN ARRAY
20 DIM S(20)
30 FOR Z = 1 TO 20
40 READ S(Z)
50 NEXT Z
60 DATA 0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,
   9,11,12
70 LET L = 20
80 LET BTM = 1
90 LET TP = L
100 INPUT"INPUT SCORE";N
110 FOR Z = 0 TO 1 STEP 0
120 LET L = TP—BTM
130 LET MD = BTM + INT(L/2)
140 IF N = S(MD) THEN LET Z = 2
150 IF N>S(MD) THEN LET BTM = MD
160 IF N<S(MD) THEN LET TP = MD
170 NEXT Z
180 PRINT"THE SCORE WAS IN ELEMENT
   NO. ";MD
190 STOP
```

VARIABLES

For variations in variable names, see previous 'Basic Flavours' (page 257).

STEP 0

In both programs in the main text there is a reference to "... THEN LET Z = X". The values for the variable X are:

Oric-1	replace X by 1
Dragon 32	replace X by 1
Lynx	replace X by 2
BBC Micro	replace X by 2
Commodore 64 and Vic-20	replace X by 1

LYNX

Both programs in the main text will run on the BBC, the Dragon 32, the Lynx, the Oric-1, the Commodore 64 and Vic-20 provided that the Basic Flavours concerning variable names and Step 0 are implemented. The Spectrum listings differ from the norm in the DIM statement in line 100 above, and in the test in line 1020 above, otherwise they may be used as a guide to implementation on the other machines. The Lynx version of line 100 above is:

```
100 DIM F$(4)(6)
```

Lynx users will have discovered that, contrary to the statement on page 137, the machine does support string arrays.

TONY LODGE