A(5) = 3, then if X has the value 1, A(X) will be the contents of A(1), which is 4. A(X + 1) will be the contents of A(2), which is 9, and so on.

Look at the program and see if you can see exactly what is going on. Line 20 sets variable N to the number of numbers we want to sort. Let's assume we want to sort five numbers: when the program is run we will type in 5 and then hit RETURN.

Line 30 is the DIMension statement. If N is 5, it sets the size of the array to 5. This line is equivalent to DIM A(5).

Lines 40 to 60 are a FOR-NEXT loop that allows us to type in the five numbers. Most versions of BASIC prompt the user with a question mark on the screen. RETURN will have to be pressed after each number has been entered. The numbers may be more than one digit, and may include decimal fractions.

Line 90 sets the variable S to 0. This variable is being used as a 'flag'. Later in the program, A is tested to see if it is 1 or not. It is only ever set to 1 if two numbers have been swapped, as we shall see in line 240. We shall investigate the use of 'flags' in more detail later in the course.

Line 100 sets up the limits for a loop; in this case from 1 to 4 (because N is 5 so N — 1 is 4). The first time through the loop, L is 1 so A(L) in line 110 will be A(1) or the first element in the array and A(L + 1) will be A(2), the second element in the array. The next time round the loop, L will be incremented to 2, so A(L) will be equivalent to A(2) and A(L + 1) will be equivalent to A(3). Line 110 tests to see if A(L) is greater than the number immediately to its right in the array. The sign for 'greater than' is >.

If the first number is bigger than the next one, the program branches to a subroutine that swaps the numbers. If the first number is not bigger than the next one, there is no branch to the subroutine and BASIC simply continues to the next line, which is the NEXT L statement. After the loop has been repeated four times, it stops the program and goes to line 130 which tests the 'swap' flag, S, to see if it has been set or not. If it has been set (in the 'swap' subroutine), the program branches back to line 90 to repeat the comparison process. If S is not 1 it means no swap took place, so all the numbers are in order. The rest of the program simply prints them out.

The swap subroutine needs a temporary variable to store one of the numbers to be swapped. After the two numbers have been swapped in lines 210, 220 and 230, the 'swap' flag S is set to 1 and then the program RETURNs to the main program.

```
10 PRINT "HOW MANY NUMBERS DO YOU WANT
TO SORT?"
20 INPUT N
30 DIM A(N)
40 FOR X = 1 TO N
45 PRINT "NEXT NUMBER"
50 INPUT A(X)
60 NEXT X
70 REM
80 REM SORT ROUTINE
90 LET S = 0
100 FOR L = 1 TO N — 1
110 IF A(L) > A(L + 1) THEN GOSUB 200
120 NEXT L
130 IF S = 1 THEN 90
140 FOR X = 1 TO N
150 PRINT "A(";X;") = ";A(X)
160 NEXT X
170 END
180 REM
190 REM
200 REM SWAP SUBROUTINE
210 LET T = A(L)
220 LET A(L) = A(L + 1)
230 LET A(L + 1) = T
240 LET S =1
250 RETURN
```

Basic Flavours

IF... THEN

If this program is to be run on the ZX81 or Spectrum, line 130 must be amended to read: 130 IF S=1 THEN GOTO 90

END

This statement is not available on the ZX81 and Spectrum, so use STOP instead

LET

In assignment statements, such as 90 LET S= 0, the word LET is optional on most machines but not on the ZX81 and Spectrum. The Lynx adds it into program listings automatically

Exercises

- Extend the program to find the average value of the numbers input. The average is equal to the sum of items divided by the total number of items. The simplest way to do this is to insert a GOSUB just before the END statement in line 170. The subroutine should read each of the elements in the array and add the values to a 'sum' variable. After all the elements have been added, the sum should be divided by the number of elements. The sum is most easily derived by using the number of elements as the upper limit of a FOR-NEXT loop.
- Change one line in the program so that the numbers will be sorted in descending order.
- This exercise is directed mainly at owners of the TI99/04A which does not like having variables used as subscripts in subscripted variables. TI BASIC does, however, accept statements such as DIM A(12). Rewrite the program so that the INPUT statement expects an exact number of numbers to be input, 12, say. This will avoid the problem of using a variable name as a subscript. Lines 100 and 110 will also have to be changed. The swap subroutine will not work in TI BASIC for the same reason. This will have to be changed too.
- A tough one. Our way of sorting numbers is by no means the only way to do it. See if you can think up an alternative method.