```
50 IF S = 0 THEN GOTO 120
60 IF S = 1 THEN GOTO 140
100 PRINT "THE NUMBER WAS NEGATIVE"
110 GOTO 999
120 PRINT "THE NUMBER WAS ZERO"
130 GOTO 999
140 PRINT "THE NUMBER WAS POSITIVE"
150 GOTO 999
999 END
```

If you look at the values 'returned' by the SGN function to S in line 30 (these are tested in lines 40, 50 and 60) you will see that there are three possible values. −1 is returned if the argument in the brackets was a negative number, 0 if the argument was zero and 1 if the argument was a positive number. Using the SGN function in line 30 saves several lines of programming. We could have written:

```
IF N < 0 THEN LET S = -1
IF N = 0 THEN LET S = 0
IF N > 0 THEN LET S = 1
```

The action performed by a BASIC function can always be achieved through normal programming; using a function just saves time, space and programming effort.

Here are a few more numeric functions. ABS returns the 'absolute' value of a number. The absolute value of a number is the same as its real value with the sign removed. Thus, the absolute value of −6 is 6. Try this:

```
10 LET X = -9
20 LET Y = ABS(X)
30 PRINT Y
40 END
```

MAX finds the maximum value of two numbers. Thus:

```
10 LET X = 9
20 LET Y = 7
30 LET Z = X MAX Y
40 PRINT Z
50 END
```

MIN is similar to MAX but finds the smaller of two numbers. Try this:

```
10 PRINT "INPUT A NUMBER"
20 INPUT X
30 PRINT "INPUT ANOTHER NUMBER"
40 INPUT Y
50 LET Z = X MIN Y
60 PRINT Z
70 END
```

Notice that these latter two functions have two arguments instead of one, and they don't need to be enclosed in brackets. Most BASICS also have a number of other numeric functions, including LOG to find the logarithm of a number, TAN to find the tangent, COS to find the cosine and SIN to find the sine. We will look at some of the ways these 'trigonometrical' functions can be used later.

BASIC also has several built-in functions that operate on character strings. We used some of these in our name-sorting program (page 135) but at the time did not look closely at how they worked. Now we will look at these and a few other string functions in more detail.

One of the most useful string functions is LEN. This counts the number of characters in a string enclosed in double quotation marks or the number of characters assigned to a string variable. Try this:

```
10 LET A$ = "COMPUTER"
20 LET N = LEN(A$)
30 PRINT "THE NUMBER OF CHARACTERS IN THE
   STRING IS ";N
40 END
```

Why would we ever need to know how many characters there are in a string variable? To see why, enter and run this short program designed to build a 'name triangle'. It prints first the first letter of a word, then the first and second letter and so on until the whole word is printed.

```
5 REM PRINTS A 'NAME TRIANGLE'
10 LET A$ = "JONES"
20 FOR L = 1 TO 5
30 LET B$ = LEFT$(A$,L)
40 PRINT B$
50 NEXT L
60 END
```

Now run this program. Can you figure out what the printout will be? It should look like this:

```
J
JO
JON
JONE
JONES
```

This short program uses the LEFT$ function to extract characters from a string. LEFT$ takes two arguments. The first specifies the string and the second (which comes after a comma) specifies the number of characters to be extracted from the string, starting from the left of the string. A$ has been assigned the string "JONES" so LEFT$(A$,1) would 'return' the letter J. LEFT$(A$,2) would return the letters JO. The short program above uses an index, L, that ranges from 1 to 5, so that the second argument in the LEFT$ function goes up from 1 to 5 each time through the loop. We knew exactly how many characters there were in the word we wanted to print (JONES), so it was easy to decide that 5 should be the upper limit in the FOR-NEXT loop. But what would we do if we did not know beforehand how many characters there would be in the loop?

This is where the LEN function comes in. LEN takes a string (in double quotes) or a string variable as its argument. Here are a few examples to show how it works:

```
10 REM PROGRAM TO TEST THE 'LEN' FUNCTION
20 PRINT LEN("COMPUTER")
30 END
```