This subroutine is nothing more than a series of PRINT statements designed to produce crude graphics on the screen. Your BASIC may well have much better screen graphics, and if this is the case it would be better to substitute the appropriate graphics statements in place of our subroutines.

Once the program has chosen a dice for you at random, it will then repeat the process to select and display a dice for the computer. The part of the program that decides who has won has been incorporated in the main program; it could just as well have been written as a subroutine, but this would hardly be worth it since it is only four lines long. Line 200 compares M (my dice) with C (computer's dice) to see if they are equal. If they are, the words IT'S A DRAW are assigned to the string variable S$. Line 210 tests to see if M is greater than C. If it is, it assigns the words YOU HAVE WON to S$. Line 220 tests to see if M is less than C. If it is, it assigns the words THE COMPUTER HAS WON to S$. Line 240 simply prints the result and the game is over. Although this program is rather long, it is essentially very simple. It uses only one function, RND, has no loops, no subscripted variables and nothing more complicated than a few IF . . . THEN statements.

Given that the RND function is so variable, and that some versions of BASIC (Microsoft's, for example) require the RANDOMIZE statement to generate a new sequence of random numbers, is there any way we could generate truly random (i.e. unpredictable) numbers without using these functions? Several techniques are available.

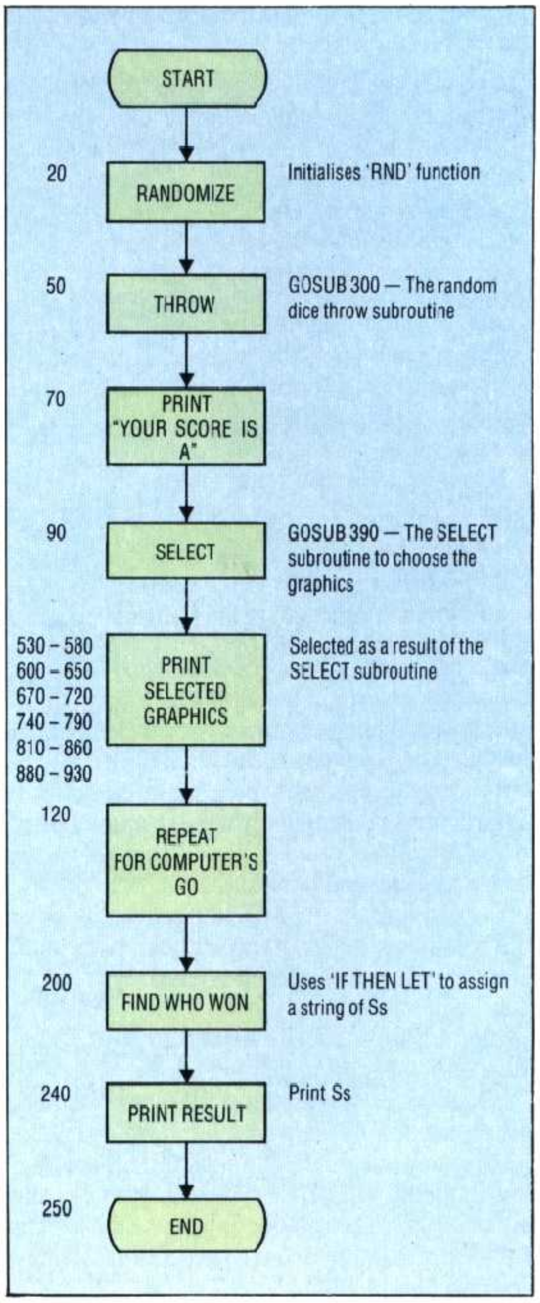
One of the functions we have not looked at so far is INKEY$ (pronounced 'inkey-string'). Each time the word INKEY$ is encountered, the program inspects the keyboard to see if a key has been pressed. The program does not wait for a character to be input as it does when the command INPUT is used. So the command INKEY$ is usually placed in a loop. The program then continually scans the keyboard, waiting for something to be input. There is usually a test within the loop to terminate it, if an appropriate character has been input. This makes it possible to write a program to form a counting loop that will terminate when a specific character has been typed in. What would happen if we used this program?

```
10 PRINT "HIT THE SPACE-BAR"
20 FOR X = 0 TO 1
30 LET R = R + 1
40 LET A$ = INKEY$
50 IF A$ = "  " THEN GOTO 80
60 LET X = 0
70 NEXT X
80 FOR Q = 0 TO 1
90 IF R < 10 THEN GOTO 130
100 LET Q = 0
110 LET R = R / 10
120 NEXT Q
130 PRINT INT (R)
140 END
```

Would R be a random number? It should be, so let's look at the program and see why.

Line 10 prints the prompt HIT THE SPACE-BAR. Before we have time to respond to this prompt, the program has entered the FOR X = 0 TO 1 loop in line 20. 0 and 1 may seem like strange limits for the loop, but we will see how this structure is used shortly. Line 30 assigns the value 1 to variable R the first time through the loop. Line 40 assigns whatever character is typed in on the keyboard to the string variable A$ in line 40. This is done using the INKEY$ function. If you were to hit the letter R, R would be assigned to A$. Line 50 tests A$ to see whether it is a space (this is represented in BASIC as a space between double quote marks thus " "). If A$ is a space, the program branches using the GOTO statement, but if A$ is not a space, the program continues to the next line.

This is line 60, which says LET X = 0. Now X is the index of the loop. The NEXT X statement in line 70 causes the program to return to the beginning of the loop in line 20. Since X has been reset to 0, the loop repeats it. In this way the FOR X = 0 TO 1 loop will be repeated indefinitely, as long as the IF A$ =



START
20
RANDOMIZE
Initialises 'RND' function
50
THROW
GOSUB 300 — The random dice throw subroutine
70
PRINT "YOUR SCORE IS A"
90
SELECT
GOSUB 390 — The SELECT subroutine to choose the graphics
530 – 580
600 – 650
670 – 720
740 – 790
810 – 860
880 – 930
PRINT SELECTED GRAPHICS
Selected as a result of the SELECT subroutine
120
REPEAT FOR COMPUTER'S GO
200
FIND WHO WON
Uses 'IF THEN LET' to assign a string of Ss
240
PRINT RESULT
Print Ss
250
END

Program Flow
The flowchart shows the main actions performed by the program in simplified form. Corresponding line numbers are given on the left and short explanatory notes on the right. This is not a full flowchart as many of the 'decisions' and branches in the program are not shown