BASIC for GREET can be tackled with little further refinement:

IV 2 (GREET) 1 (display message) BASIC CODE

```
REM *GREET* SUBROUTINE
PRINT
PRINT
PRINT
PRINT
PRINT TAB(12);"*WELCOME TO THE*"
PRINT TAB(9);"*HOME COMPUTER COURSE*"
PRINT TAB(6);"*COMPUTERISED ADDRESS
   BOOK*"
PRINT
PRINT TAB(5);"(PRESS SPACE BAR TO CONTINUE)"
```

V 2 (GREET) 2 (LOOP wait for space bar) BASIC CODE

```
LET L = 0
FOR L = 1 TO 1
IF INKEY$ <> "   " THEN LET L = 0
NEXT L
```

IV 2 (GREET) 3 (call *CHOOSE*) BASIC CODE

```
GOSUB *CHOOSE*
RETURN
```

Notice that we have now started to initialise variables in the various routines that we write, by using statements of the form LET I = 0. Strictly speaking, this is unnecessary in some of the circumstances in which we have used it. Nevertheless, it is a good habit to get into if you can remember, and if you have enough RAM space available. There are three reasons: first because having a list of LET statements at the start of any routine serves as a useful reminder of what local variables that routine uses. Secondly, because you cannot be sure of what was left in a variable from the last time it was used in a routine (though this does not always matter). Thirdly, as we shall be explaining to you later in the course, putting in statements of the form LET I = 0 in the right order can speed up the execution of a program.

We have changed the way in which we use a FOR...NEXT loop to simulate a DO...WHILE or REPEAT...UNTIL structure from previous instalments of the course. Instead of using FOR I = 0 TO 1 or FOR I = 0 to 1 STEP 0, we are now using FOR I = 1 to 1. This will run correctly on all the home computers we regularly cover, where the other methods required 'Basic Flavours' for various machines. FOR I = 1 TO 1...NEXT I will execute the loop just once. However, if anywhere in the body of the loop I is set to 0 then the loop will execute again, and so on. We can either insert a LET I = 0 statement as the result of an exit condition failing or we can set I to 0 immediately after the FOR statement, and set it to 1 if the exit condition succeeds. Thus, both the following loops achieve the same objective:

```
FOR I = 1 TO 1
IF INKEY$ <> "   " THEN LET I = 0
NEXT I
```

or

```
FOR I = 1 TO 1
LET I = 0
IF INKEY$ = "   " THEN LET I = 1
NEXT I
```

The BASIC code we have just produced is all that is needed for the complete GREET block in the main program. We haven't put in line numbers because we can't really do that until all the program modules are ready for final coding. For instance, we do not know at this stage what the appropriate line numbers are for the GOSUB commands. If you want to test the module at this stage, it will be necessary to create some dummy inputs and dummy subroutines. Some points to note about this program fragment are the use of the TAB function and the 'clear screen' statements. TAB moves the cursor along the line by the number (the 'argument') specified in the brackets. The numbers we have given will print the message neatly centred in a screen 40 characters wide. If your display has less than this (for example, the Spectrum displays 32 characters per line) or more (larger computers usually display 80 characters), these TAB arguments will need to be altered accordingly. The instruction to clear the screen in many versions of BASIC is CLS, but the version of Microsoft BASIC used to develop this program does not support this. Instead, we have used PRINT CHR$(12), since our machine uses ASCII 12 as its 'clear screen' non-printable character — others commonly use ASCII 24 for the same function.

```
10 REM DUMMY MAIN PROGRAM
20 PRINT CHR$(12)
30 GOSUB 100
40 END
100 REM *GREET* SUBROUTINE
110 PRINT
120 PRINT
130 PRINT
140 PRINT
150 PRINT TAB(12);"*WELCOME TO THE*"
160 PRINT TAB(9);"*HOME COMPUTER COURSE*"
170 PRINT TAB(6);"*COMPUTERISED ADDRESS
    BOOK*"
180 PRINT
190 PRINT TAB(5);"(PRESS SPACE BAR TO
    CONTINUE)"
195 LET L = 0
200 FOR L = 1 TO 1
210 IF INKEY$ <> "   " THEN LET L = 0
220 NEXT L
230 PRINT CHR$(12)
240 GOSUB 1000
250 RETURN
1000 REM DUMMY SUBROUTINE
1010 PRINT "DUMMY SUBROUTINE"
1020 RETURN
```

We will now use exactly the same approach to refine the CHOOSE procedure.

II 3 (CHOOSE)

```
BEGIN
   1. PRINT menu
```