Luckily, SIZE is always one bigger in value than the highest valid record. In other words, there is no record at position SIZE in the arrays, so this fix will not modify any existing record. But without some extensive REMs explaining what's going on, think how confusing these three lines would be to someone who had not been involved in the development of the program!

Back to the more interesting problem of dealing with 'near misses'. Suppose we had entered someone's name as Pete Jones during an *ADDREC* operation, but as Peter Jones during *FNDREC*. These would be converted to the standardised forms JONES PETE and JONES PETER respectively, and no match would be found during the search, even though the record we wanted was there. We will not attempt to solve this problem, because a satisfactory solution would represent a major programming task. For readers interested in experimenting, however, here are some pointers:

```
BEGIN {search array for exact match}
  IF exact match found
    THEN PRINT full record
    ELSE search array for close-match
      IF close-match found
        THEN PRINT record for close-match
        ELSE PRINT "NO RECORD FOUND"
      ENDIF
  ENDIF
END
```

The procedure for close-match could be something along the lines of:

```
BEGIN {close-match}
  Search array for exact surname match
  IF exact surname match
    THEN search forenames for max-match
    PRINT record for max-match
    ELSE search surnames for max-match
      IF surname max-match found
        THEN PRINT record for max-match
      ENDIF
  ENDIF
END
```

The procedure for max-match could be roughly defined as finding the target string with the maximum number of characters in common with those in the key string. Or it could accept a situation in which the key string was wholly contained within the target string, or vice versa. There are no simple solutions, but plenty of scope for enterprising programming.

There is one 'side effect' of the program fragment presented. Suppose the following sequence of events takes place. There are ten records in the data file. You run the program and then use *ADDREC* to add a new record, followed by *FNDREC* to locate a record. When *EXPROG* is finally run, to save the file and terminate the program, the record you added will not be saved (although all the other records will be). This is a direct result of something that happened in the execution of *FNDREC*. Can you see why the record added will not be saved?

In the next instalment of the course we will explain how to prevent this loss of data; show what the CURR variable is used for, and describe how to delete or modify a record. Other options on the main menu (*FNDTWN* etc.) are closely similar to routines we have already worked out. Readers will be left to implement them for themselves if they are required.

Finally, consider what would happen if there were exactly 50 records in the data file and the modified *FNDREC* routine (that calls *MODNAM*) were used. (Hint: SIZE will have the value 51.)

Basic Flavours

For Sinclair machines, the following modifications are required:

```
13000 REM *FNDREC* TEST VERSION
13010 IF RMOD = 1 THEN GOSUB 11200
13020 PRINT "INPUT KEY"
13030 INPUT SS
13100 LET BTM = 1
13110 LET TOP = SIZE - 1
13120 FOR L = 1 TO 1
13130 LET MID = INT((BTM+TOP)/2)
13140 IF MS(MID) <> SS THEN LET L=0
13150 IF MS(MID) < SS THEN LET BTM =
      MID + 1
13160 IF MS(MID) > SS THEN LET TOP = MID
      - 1
13170 IF BTM > TOP THEN LET L = 1
13180 NEXT L
:
:
13200 IF BTM > TOP THEN PRINT "RECORD
      NOT FOUND"
13210 IF BTM <= TOP THEN PRINT "RECORD
      IS NO ";MID
:
:
13240 STOP
13250 RETURN
```

Notice once again the problem of single-letter string variable names: here SS and MS have been substituted for SCHKEYS, MODFLDS

Erratum
In the ZX81 and Spectrum Basic Flavours on page 257, lines 9990 to 9992 should not have been included in Step 3

```
13000 REM VERSION OF *FNDREC* FOR TESTING
13010 IF RMOD = 1 THEN GOSUB 11200
13020 INPUT "INPUT KEY ";SCHKEY$
13030 REM
13040 REM
13050 REM
13060 REM
13070 REM
13080 REM
13090 REM
13100 LET BTM = 1
13110 LET TOP = SIZE - 1
13120 FOR L = 1 TO 1
13130 LET MID = INT((BTM + TOP)/2)
13140 IF MODFLD$(MID) <> SCHKEY$ THEN L = 0
13150 IF MODFLD$(MID) < SCHKEY$ THEN BTM = MID + 1
13160 IF MODFLD$(MID) > SCHKEY$ THEN TOP = MID - 1
13170 IF BTM > TOP THEN L = 1
13180 NEXT L
13190 REM
13200 IF BTM > TOP THEN PRINT "RECORD NOT FOUND"
13210 IF BTM <= TOP THEN PRINT "RECORD IS NO ";MID
13220 REM
13230 REM
13240 STOP
13250 RETURN
```

TONY LODGE