while the first gets only three out of six. On the other hand, the first has three letters in correct sequence, while the second has only two.

The choice is largely arbitrary. We will opt for giving priority to an exact match between KEY$ and a substring of the name in memory. If no exact match with a substring can be found, the program will try to get the largest number of common letters. Here's the program stated in terms of input and output:

INPUT
A character string
OUTPUT
The closest match to the input string

The following program, in a pseudo-language close to BASIC, will search through the strings in an array and examine the first 'n' letters in each, where 'n' is the number of letters in the key (KEY$). If there is no match, a message to that effect will be printed:

```
DIM ARRAY$(4)
FOR L = 1 TO 4
READ ARRAY$(L)
NEXT L
DATA "ROBERT", "RICHARD", "ROBIANA",
   "ROBERTA"
LET KEY$ = "RON"
LET LKEY = LEN(KEY$)
LET SEARCHING = 0
LOOP FOR INDEX = 1 TO 4
   IF KEY$ = LEFT$(ARRAY$(INDEX),LKEY)
      THEN PRINT "MATCH IS ";ARRAY$(INDEX)
         LET SEARCHING = INDEX
   ENDIF
ENDLOOP
IF SEARCHING = 0
   THEN PRINT KEY$; "IS NOT AN EXACT MATCH
        OF ANY"
      PRINT "FIRST ";LKEY; "CHARACTERS"
```

After this, the program could go on to look at groups of characters LKEY long, starting with the second character in each string. If none of these matches, groups starting with the third character could be searched, and so on. Finally, if none of the triplets of characters in the strings matches, the program could try to find which string had the largest number of letters in common with KEY$. This is left as an exercise for the reader.

We could in fact write pages on the subject of 'fuzzy' matching, and the different techniques employed in commercial database packages. Most offer the ability to search on the first few characters in the field, like the code we have just been developing. Others will retrieve a record if the specified sequence of characters appears anywhere in the field, or indeed anywhere in the record. A 'wildcard' facility is particularly useful, so that specifying: J?N would find JONES, or JANE but not JOHN. The most sophisticated form of fuzzy matching works phonetically, so that entering SMITH would also find SMYTHE.

Basic Flavours

ZX81 SPECTRUM

This is the listing of the program to reverse the order of Firstname and Surname, first published on page 136:

```
100 CLS
200 PRINT "ENTER NAME IN THE FORM"
300 PRINT "FIRSTNAME SURNAME"
400 PRINT "E.G. JILL THOMPSON"
500 INPUT "ENTER NAME";N$
600 GOSUB 9500
700 PRINT "NAME IN STANDARD FORM IS"
800 PRINT N$
1000 STOP
9500 REM S/R TO REVERSE NAME ORDER
9520 GOSUB 9600
9540 IF P=0 THEN RETURN
9560 LET N$=S$+",  "+F$
9580 RETURN
9600 REM S/R TO SLICE N$ AT A SPACE
9620 LET N=LEN (N$)
9630 LET P=0
9640 FOR K=1 TO N
9650 IF N$ (K)=" " THEN LET P=K
9655 IF N$ (K)=" " THEN LET K=N
9660 NEXT K
9670 IF P=0 THEN RETURN
9680 LET F$=N$( TO P-1)
9700 LET S$=N$(P+1 TO )
9720 RETURN
```

LEFT$
RIGHT$
MID$

On the Commodore 64, Vic-20, Oric-1, and Lynx, replace lines 9600 to 9720 of the Spectrum listing by these lines:

```
9600 REM S/R TO SLICE N$ AT A SPACE
9620 LET N=LEN (N$)
9630 LET P=0
9640 FOR K=1 TO N
9650 IF MID$(N$,K,1)="  " THEN LET P=K:
     LET K=N
9660 NEXT K
9670 IF P=0 THEN RETURN
9680 LET F$=LEFT$(N$,P-1)
9700 LET S$=RIGHT$(N$,N-P)
9720 RETURN
```

INSTR

On the Dragon 32 and the BBC Micro, replace lines 9600 to 9720 of the Spectrum listing by these lines:

```
9600 REM S/R TO SLICE N$ AT A SPACE
9620 LET N=LEN (N$)
9640 LET P=INSTR(N$, "  ")
9670 IF P=0 THEN RETURN
9680 LET F$=LEFT$(N$,P-1)
9700 LET S$=RIGHT$(N$,N-P)
9720 RETURN
```

As we have mentioned before, INSTR is a useful function, particularly when dealing with database-type applications such as this. If your machine has INSTR, then you may like to attempt a more sophisticated form of 'fuzzy' matching.

INPUT

On the BBC Micro, replace line 500 of the Spectrum listing by:

```
500 INPUT "ENTER NAME", N$
```