This program fragment in pseudo-language is now close enough to a programming language to be coded. Our version of Microsoft BASIC does not allow the names of string variables to be full words, so single letters will be used instead. Thus NAMES$ becomes N$.

```
1000 REM UPPER CASE CONVERSION SUBROUTINE
1010 INPUT "INPUT NAME"; N$: REM FOR TESTING
       ONLY
1020 LET P$ = "": REM ENSURE STRING IS EMPTY
1030 FOR L = 1 TO LEN(N$): REM INDEX OF LOOP
1040 LET T$ = MID$(N$,L,1): REM EXTRACTS
       CHARACTER
1050 LET T = ASC(T$): REM FINDS ASCII VALUE OF
       CH.
1060 IF T >= 97 THEN LET T = T - 32: REM U/C
       CONV.
1070 LET T$ = CHR$(T)
1080 LET P$ = P$ + T$: REM P$ IS TEMPSTRING
1090 NEXT L: REM END OF LOOP
1100 LET N$ = P$: REM N$ IS NOW ALL UPPER CASE
2000 PRINT N$: REM FOR TESTING ONLY
2010 END: REM FOR TESTING ONLY. SUBSTITUTE
2020 REM 'RETURN' IN ACTUAL PROGRAM
```

This program fragment would normally be used as a subroutine to be called from a main program. We have written it for testing purposes with an INPUT statement, a PRINT statement, lots of REM statements, and an END. These would be removed before the subroutine was incorporated into a full program.

Basic Flavours

LYNX

The Lynx has a function, UPCS(A$), that converts the letters in A$ to upper case, so the program reduces to:

```
1010 INPUT "input name";N$
1020 LET N$=UPCS(N$)
2000 PRINT N$
2010 REM "RETURN" HERE IN ACTUAL
       PROGRAM
```

SPECTRUM

The Spectrum program in full is:

```
1010 INPUT "INPUT NAME";N$
1020 LET P$=""
1030 FOR L=1 TO  LEN  N$
1040 LET T$=N$(L)
1050 LET T= CODE T$
1060 IF T>=97 THEN LET T=T—32
1070 LET T$=CHR$ T
1080 LET P$=P$+T$
1090 NEXT L
2000 LET N$=P$
2010 PRINT N$
2020 REM "RETURN" HERE
       IN ACTUAL PROGRAM
```

DRAGON 32

This program will run on the Dragon, but lower case characters are reserved for the printer, so the problem doesn't really arise

BBC Micro

Replace line 1010 with:

```
1010 INPUT "input name", N$
```

Exercises

- Refine all the stages above to the point where they could be turned into a BASIC program. The 'pseudo-language' you use does not need to be the same as ours, but it is a good idea to follow the convention of using upper case letters for keywords that will probably correspond to statement words in the target language (e.g. LOOP, IF, LET etc.). Use small letters for operations that will need to be stated more explicitly when finally coded. These can be in ordinary English.
- Having developed the programs to a satisfactory level of refinement, convert them into program modules (subroutines) in BASIC. Test them individually using dummy inputs and print statements that can be removed later if they work properly.

Revision Exercises

These exercises demonstrate most of the commonly used statements and functions in BASIC. There are no 'trick' questions and nothing new is being introduced. If you can do all or most of these exercises without difficulty you may consider yourself on the way to becoming a fully-fledged BASIC programmer.

- Write a program to accept two numbers input from the keyboard, to add them and to print out the result.
- Assign two words (character strings) to two string variables and then create a third string variable that concatenates (joins together) the two original words. Print the third variable.
- Write a program that allows you to type in any word on the keyboard and then prints the length of the string in the messageTHE WORD YOU TYPED HAS * CHARACTERS (* stands for a number).
- Write a program that accepts a single character typed in at the keyboard and then tells you what the ASCII value of the character (in decimal) is.
- Write a program that prompts you with the message TYPE IN A WORD and then answers with the message THE LAST LETTER OF THE WORD WAS *(* stands for a letter).
- Write a program that prompts you to TYPE IN A PERSON'S NAME and then responds with THE SPACE WAS THE *TH CHARACTER.
- How would you modify the program above to print 2ND instead of 2TH if the space was in the second position?
- Write a program that prompts you to TYPE IN A SENTENCE and then responds with the message THE SENTENCE YOU TYPED HAS * WORDS (assume that there will be one word more than there are spaces in the sentence).
- Test your computer to see if characters (or special graphics) have been assigned to ASCII values from 128 to 255 (use a loop and the CHR$(X) function).