Imperial College of Science, Technology and Medicine

Department of Computing

M.Sc. C++ Programming – Unassessed Exercise No. 4

Issued: Friday 16 October 2015

Problem Description

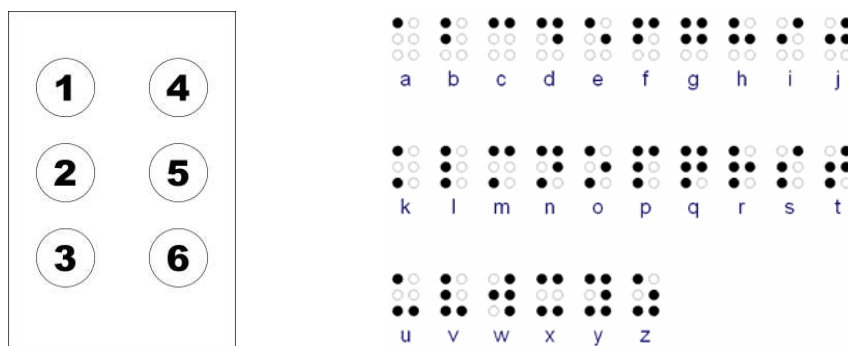The *Braille* system[1] is used by blind people to read and write.

Figure 1: A Braille cell (left) and Braille letter encodings (right).

Braille text consists of a sequence of *cells*, each of which contains six dot positions arranged in two columns of three. The dot positions are numbered 1,2,3 downwards from the top of the lefthand column and 4,5,6 from the top of the righthand column (see left of Figure 1).

The presence or absence of raised dots on the dot positions gives the coding for a symbol. For example, letters are encoded as shown on the right in Figure 1. By default, letters are lower case.

Capital letters are indicated by a preceding each letter by the *capital sign* .

Numbers are represented by preceding each digit in the number by the *number sign*[2] . The digits themselves are encoded by reusing the letters `a` through `i` to represent the digits `1` through `9` respectively, and using `j` to represent the digit `0`. So we encode `20` as .

Punctuation is encoded as follows. A period/full stop ('.') is , a comma (',') is , a semicolon (';') is , a hyphen/dash ('-') is , an exclamation mark ('!') is and a question mark ('?') is . Opening and closing brackets ('(' and ')') are both encoded as .

---

[1] Devised by Louis Braille in 1821, the Braille system is an adaption of an early tactile military communications system devised by Charles Barbier called *night writing*. Night writing encodes symbols as columns of raised dots and was proposed as a way to allow Napoleon's soldiers to communicate silently and without light at night.

[2] A simplification for this exercise; in real Braille the number sign stays in effect until the next space.

Specific Tasks

1. Write a function `encode_character(ch,braille)` which produces a string (the second parameter `braille`) that represents the Braille encoding of a single input character (the first parameter `ch`). In the string encoding, a Braille cell should be represented as 6 consecutive characters, one for each dot position; raised dots should be encoded as 'O' and unraised dots as '.' For example, the character 'n' should be encoded as the string `"O.OOO."`. Any character for which the encoding has not been given (including the space character) should be encoded as `"......"`. The function should return the length of the Braille-encoded string.

   For example, the code:

   ```
   char braille[20];
   int size;
   size = encode_character('t', braille);
   ```

   should result in the string `braille` having the value `".OOOO."` and `size` having the value `6`.

   As another example, the code:

   ```
   char braille[20];
   int size;
   size = encode_character('Z', braille);
   ```

   should result in `braille` having the value `".....O|O.O.OO"` (i.e. the capital sign followed by the letter `z`)[3] and `size` having the value `12`.

   As a final example, the code:

   ```
   char braille[20];
   int size;
   size = encode_character('5', braille);
   ```

   should result in `braille` having the value `"..OOOO|O...O."` (i.e. the number sign followed by the encoding for the letter `e`)[3] and `size` having the value `12`.

2. Write a **recursive** function `encode(plaintext,braille)` which produces the Braille encoding of a plaintext input string. The first parameter to the function (i.e. `plaintext`) is an input string containing the string to be encoded. The second parameter (i.e. `braille`) is an output parameter which should contain the corresponding Braille-encoded string.

   For example, the code:

   ```
   char braille[100];
   encode("Hello!", braille);
   ```

   should result in the string `braille` having the value:

   `".....O|OO..O.|O...O.|OOO...|OOO...|O.O.O.|.OO.O."`.

   As in Question 1, the dashed vertical bars are included here only to aid understanding; they should **not** be included in the output string.

3. Write a function `print_braille(plaintext,output)` which takes a plaintext string and writes the corresponding sequence of 3x2 Braille cells to an output stream. The function should also display the corresponding plaintext character under each Braille cell (where

---

[3]Note the dashed vertical bars in the output string are included here only as an aid to understanding; they should not be included in the output string.

applicable). The first parameter (i.e. `braille`) is the input plaintext string and the second parameter is the output stream (e.g. `cout` or a file output stream).

For example, the code:

`print_braille("Hello!", cout);`.

should result in the following output written to `cout` (i.e. displayed on the screen):

```
.. O. O. O. O. O. ..
.. OO .O O. O. .O OO
.O .. .. O. O. O. O.
   H  e  l  l  o  !
```

What To Hand In

Place your function implementations in the file **braille.cpp** and corresponding function declarations in the file **braille.h**. Use the file **main.cpp**, available at:

`http://www.doc.ic.ac.uk/~wjk/C++Intro/braille/main.cpp`

to test your functions. Create a **makefile** which compiles your submission into an executable file called **braille**. If this was an assessed exercise, you would be required to submit the three files: **braille.h**, **braille.cpp** and **makefile** electronically using the CATE system.

How You Will Be Marked

You will be assigned a mark (for all your programming assignments) according to whether your program works or not, whether your program is clearly set out with adequate blank space, comments and indentation, whether you have used meaningful names for variables and functions, and whether you have used a clear, appropriate and logical design.

Hints

1. Feel free to define any auxiliary functions which would help to make your code more elegant.

2. Try to attempt all questions. If you cannot get one of the questions to work, try another.

3. The standard header `<cctype>` contains some library functions that you may find useful. In particular:

   - `int isalpha(char ch)` returns nonzero if `ch` is a letter from 'A' to 'Z' or a letter from 'a' to 'z'.
   - `int isupper(char ch)` returns nonzero if `ch` is a letter from 'A' to 'Z'.
   - `char tolower(char ch)` returns the lower case character corresponding to `ch`.
   - `int isdigit(char ch)` returns nonzero if `ch` is a digit between '0' and '9'.
   - `int ispunct(char ch)` returns nonzero if `ch` is a punctuation character.

4. Please note that your solution to **Question 2** should be **recursive**, i.e. the function should call itself. If you cannot implement a recursive solution, up to 75% of the marks will be awarded for an iterative (non-recursive) solution.