Imperial College of Science, Technology and Medicine
Department of Computing

M.Sc. C++ Programming – Unassessed Exercise No. 2

Issued: Friday 13 October 2017

Objective

You are required to implement C/C++ functions for encoding and manipulating surnames according to the way they sound, by using the *Soundex* coding system.

The Soundex coding system

The *Soundex* coding system represents surnames according to the way they sound rather than the way they are spelt. Surnames that sound the same but are spelt differently (e.g. PAYNE and PAINE) have the same Soundex code.

All Soundex codes take the form of an upper case letter followed by three digits (e.g. B652, E255 or M263). The rules for mapping a surname onto its corresponding Soundex code are as follows:

1. The letter at the start of the Soundex code is always the first letter of the surname.

2. The digits of the Soundex code are assigned using the remaining letters of the surname according to the mapping:

b, f, p, v	→	1	l	→	4
c, g, j, k, q, s, x, z	→	2	m, n	→	5
d, t	→	3	r	→	6

   All occurrences of the letters a, e, h, i, o, u, w and y are ignored. If there are not enough digits to make up a four character code, zeroes are added to the end. Additional letters are disregarded.

   For example, **Washington** is coded **W252** (W, 2 for the S, 5 for the N, 2 for the G, remaining letters disregarded). **Lee** is coded **L000** (L, 000 added).

3. If the *surname* has two or more *adjacent* letters that have the same number in the Soundex coding, they should be treated as one letter.

   For example, **Jackson** is coded as **J250** (J, 2 for the C, K ignored, S ignored, 5 for the N, 0 added).

   Note that this rule does **not** imply that it is impossible for adjacent repeated digits to occur in a Soundex coding.

Tasks

Your specific tasks are to:

1. Write a function `encode(surname, soundex)` which produces the Soundex encoding corresponding to a given surname. The first parameter to the function (i.e. `surname`) is an input string containing the surname that should be encoded. The second parameter (i.e. `soundex`) is an output string into which the Soundex encoding for the string should be placed.

   For example, the code:

   ```
   char soundex[5];
   encode("Jackson", soundex)
   ```

   should result in the string `soundex` having the value `"J250"`

2. Write a **recursive** function `compare(one, two)` which compares two Soundex codes (where `one` and `two` are strings representing the Soundex codes being compared). If the Soundex codes are the same, the function should return 1, else it should return 0.

   For example, `compare("S250","S255")` should return `0` while `compare("W252","W252")` should return `1`.

3. Write a function `count(surname, sentence)` which returns the number of words in a given sentence that have the same Soundex encoding as the given surname.

   For example, the function call

   `count("Leeson", "Linnings, Leasonne, Lesson and Lemon.")`

   should return the value `2`.

Place your function implementations in the file **soundex.cpp** and corresponding function declarations in the file **soundex.h**. Use the file **main.cpp** to test your functions. **main.cpp** can be found on the web at `http://www.doc.ic.ac.uk/~wjk/C++Intro/soundex/main.cpp`.

What to hand in

If this was an assessed exercise, you would be required to submit the files `soundex.h`, `soundex.cpp` and a `makefile` using a web submission page. But since this exercise is not assessed you do not actually need to hand anything in.

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. 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 character between `'A'` and `'Z'` or between `'a'` and `'z'`.
   - `char toupper(char ch)` returns the upper case equivalent of character `ch`.
3. There are at least three different ways of implementing your answer to Question 1. What are they? Which way is the most elegant?
4. Your answer to Question 3 (i.e. your implementation of the `count` function) should not modify the second parameter (i.e. `sentence`) directly. However, it is acceptable if your code makes and then modifies a *copy* of this parameter.