Imperial College of Science, Technology and Medicine
Department of Computing

M.Sc. C++ Programming – Unassessed Exercise No. 3

Issued: Friday 15 October 2021

Objective

A **palindrome** is a word or phrase which contains the same sequence of letters when read forwards or backwards.

Characters other than letters (including spaces) are ignored and the case of letters (i.e. whether they are upper case i.e. `A-Z` or lower case i.e. `a-z`) is irrelevant. An empty string and a string containing a single letter are both palindromes (by definition).

For example,

- The words "`rotor`" and "`Radar`" are palindromes.
- The sentence "`A man, a plan, a canal, Panama!`" is a palindrome, and so is "`Mr. Owl ate my metal worm.`"
- The strings "`a. ..`" and "`.. ..!`" are palindromes.

An **anagram** is a word or phrase formed by reordering the *letters* of another word or phrase. Again, characters other than letters (including spaces) are ignored and the case of letters is irrelevant.

For example:

- "`stain`" is an anagram of "`satin`"
- "`I am a weakish speller!`" is an anagram of "`William Shakespeare`"
- "`Here come dots...`" is an anagram of "`The Morse Code`".

The purpose of this exercise is to write a set of functions which can be used to identify palindromes and anagrams.

Tasks

Your specific tasks are to:

1. Write a function `reverse(str1, str2)` which copies `str1` into `str2` backwards. Here `str1` is an input parameter containing the string to be reversed, and `str2` is an output parameter containing the reversed string. So the code:

   ```
   char reversed[9];
   reverse("lairepmi", reversed);
   ```

   should result in the string `reversed` having the value "`imperial`". You may assume that the memory for `string2` has been allocated and is sufficient to hold the output string (including sentinel character).

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

   For example, `compare("This and that", "this, and THAT!")` should return `1`
   while `compare("these are not the SAME", "these are the SAME")` should return `0`.

3. Write a function `palindrome(sentence)` which returns `1` if the given sentence is a palindrome, and `0` otherwise. You may not make any assumptions about the size of the input sentence.

   For example, the function call `palindrome("Madam, I'm Adam")` should return `1` while the function call `palindrome("Madam, I'm not Adam")` should return `0`.

4. Write a function `anagram(str1, str2)` which returns `1` if string `str1` is an anagram of string `str2`, and `0` otherwise.

   For example, the function call `anagram("William Shakespeare", "I am a weakish speller!")` should return `1`, while the call `anagram("William Shakespeare", "I am a good speller!")` should return `0`.

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

(The four parts carry, respectively, 20%, 30%, 20% and 30% of the marks)

What To Hand In

If this was an assessed exercise, you would be required to submit three files: `words.h`, `words.cpp` and a `makefile`. But since it is unassessed, you do not need to hand in anything.

How You Will Be Marked

If this was assessed, you would be assigned a mark according to:

- whether your program works or not,
- whether your program is clearly set out with adequate blank space and indentation,
- whether your program is adequately commented,
- whether you have used meaningful names for variables and functions, and
- whether you have used a clear, appropriate and logical design.

Hints

1. Try to attempt all questions. Note that Questions 1 and 2 can be attempted independently, and if you have function prototypes for Questions 1 and 2, you can write Question 3. If you have the function prototype for Question 2, you can write Question 4.
2. The standard header `<cctype>` contains some library functions that you may find useful when answering **Question 2**. 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. Please note that your solution to **Question 2** should be **recursive**, i.e. the function should call itself. If you cannot manage a recursive solution, up to 50% of the marks will be awarded for an iterative (non-recursive) solution.
4. Feel free to define any auxiliary functions which would help to make your code more elegant. In particular, when answering **Question 4**, you might find it useful to create an auxiliary case-insensitive string sorting function.