Program #3 – Due 6 November

CSC 150

You’ve been hired by the Palindromes-R-Us company to create an application for their customers which will fulfill all their palindrome needs. In your new hire packet you find out that a palindrome is a word, line, verse, number, sentence, etc. that reads the same backward as forward. For example :

1. Was it a car or a cat I saw
2. Never odd or even
3. A man, a plan, a canal, Panama

[Note : In the examples above, the spaces, punctuation, and capitalization were added to enhance readability. The actual palindromes are : wasitacaroracatisaw, neveroddoreven, and amanaplanacanalpanama respectively]

The application you are to create provides 4 tools. Palindrome recognition, palindrome decorruption (a neologism created by Palindromes-R-Us for this new product), palindrome detective, and palindrome ciphering.

The first tool : Palindrome Recognizer® will accept a string of characters and report whether or not that string represents a palindrome. Examples (text in **bold** is user input):

Palindrome Recognizer

Enter string : **abba**

abba is a palindrome. [and a great 70’s-80’s rock group]

Enter string : **Never odd or even**

Never odd or even is a palindrome.

Enter string : **accurate**

accurate is not a palindrome.

[Note: In all the examples multiple examples shown are for demonstration only, your code should ask for one string, report the results, and return to the main menu]

Your tool will read in an entire line of text (not just up to the first space). It will then remove the spaces, punctuation, and normalize the case of the letters (i.e. make them all upper or all lower case). It will then check to see if the resulting string is a palindrome. If so, print out <original string> is a palindrome. If the resulting string is not a palindrome, print out <original string> is not a palindrome.

The second tool, Palindrome Decorruptionator™ is needed by our customers because our competitors (Palindromes-R-Us-Too) have been hacking into our systems and corrupting our best palindromes by inserting random characters into them. Fortunately, our security software is fast enough so that it detects and stops the intrusion after only a single character corruption takes place. Unfortunately, we do not know how many of our most valuable palindromes have been corrupted. Your tool will accept a string of characters and report whether or not it’s a palindrome and, if not, see if it is a palindrome that suffers corruption from a single errant character. Examples :

Palindrome Decorruptionator

Enter string : **abbac**

abbac is not a palindrome.

abbac was corrupted! It is the palindrome abba

Enter string : **Was itt a car or a cat I saw**

Was itt a car or a cat I saw is not a palindrome.

Was itt a car or a cat I saw was corrupted! It is the palindrome wasitacarorcatisaw

Enter string : **accurate**

accurate is not a palindrome.

accurate is not a corrupted palindrome.

[Note: Multiple examples shown are for demonstration only, your code should ask for one string, report the results, and return to the main menu]

Note that the Decorruptionator™ does not have to replace spacing, punctuation, and capitalization for the corrupted palindrome, we have other agents working on tools to provide that service. Since you should already have a function that recognizes palindromes, you can use it in your Decorruptionator function to check the series of strings created by removing one character at a time from the string entered by the user. This shows the benefit of writing reuseable, single purpose functions that can be called when a specific task needs to be performed.

A second competitor (We-R-Palindromes-As-Well) has also been hacking into our systems but their malicious software isn’t able to insert random characters. Their software scrambles the letters of our palindromes rendering them almost worthless. The third tool, Palindrome Detective© will analyze character strings to discover if the letters could possibly have been a palindrome. Examples :

Palindrome Detective

Enter string : **aabb**

aabb is a potential palindrome.

Enter string : **aa aiI a aaWsws ccot t**

aa aiI a aaWsws ccot t is a potential palindrome.

Enter string : **accurate**

accurate is not a potential palindrome

Our research staff have discovered that potential palindromes can be found by counting the occurrence of letters. If individual letters appear an even number of times in the string then the characters make up a potential palindrome. Additionally, if a single character appears an odd number of times, the string is still a potential palindrome since a single character can appear in the middle of the palindrome (thus matching itself. Example : abcd**e**dcba). Note that the Palindrome Detective© does not have to create the palindrome, simply detect if the string has the makings of a palindrome. We have other agents working on tools to provide that service.

Finally, our board of directors has decided that to protect our valuable palindromes in the future we need to use a Palindromic Cipher to hide our palindromes from the malicious software. A Palindromic Cipher is a technique that involves shifting (or rotating) every other character in a string a fixed number of positions. Examples :

Palindromic Cipher

Enter string to encode : **abba**

Enter rotation distance (1-25): **4**

Palindromic Cipher complete : wbxa

Enter string to encode : **Was it a car or a cat I saw**

Enter rotation distance (1-25) : **5**

Palindromic Cipher complete : ranioaxamomaxaoinar

Enter string to encode : **accurate**

accurate is not a palindrome

Palindromic Decipher

Enter string to decode : **wbxa**

Enter rotation distance (1-25): **4**

Palindromic Decipher Text = abba

Enter string to decode : **ranioaxamomaxaoinar**

Enter rotation distance (1-25): **5**

Palindromic Decipher Text : wasitacaroracatisaw

Here is an example of how your main menu should look:

1. Palindrome Recognizer

2. Palindrome Decorruptionator

3. Palindrome Detective

4. Palindromic Cipher

5. Palindromic DeCipher

6. Quit

Enter Choice :

# Input constraints

* Strings will not exceed 80 characters in length.
* Only alphabetic characters should be considered. Your code should remove punctuation, spaces, digits, etc. before processing.
* Your program should check input against invalid menu selections and rotation distances.
* Note that the Cipher should not rotate non-palindromes.
* You are **not allowed** to use string library functions like **strrev** to perform string manipulations. You are allowed to use library functions like strlen, strcpy\_s, and other basic string utility functions.

**Timeline:**

|  |  |
| --- | --- |
| 10/14: | You should have the menu completed and be able to get user input. |
| 10/19: | Your code should be able to clean up input strings and detect palindromes |
| 10/26: | Two or three of the four tools should be working and tested with a variety of inputs. |
| 10/30: | All four tools working and Doxygen documentation well underway. |
| 11/4: | Final testing underway, Doxygen comments complete, internal comments finalized |
| 11/6: | Submit finished program. |
|  |  |
|  |  |

**Comments and suggestions.**

* + You have a lot of time to work on this program. Do not let that lull you into thinking you can put off starting on it.
  + Do not try and write the entire program all at one time. Work on the program in small sections, as the timeline indicates. Use drivers and stubs to work on manageable portions.
  + Debugging Hint: Make sure your program’s output matches the sample run above and from the running example file.
  + Be sure to follow the coding style guidelines that can be found on the class website at [www.mcs.sdsmt.edu/csc150/Course.](http://www.mcs.sdsmt.edu/csc150/Course/)
  + Name your code file **prog3.cpp**. Points will be deducted if this naming convention is not followed.
  + Your program must correctly compile in Visual C++ 2013.
  + Be sure your code file is readable and neat. Do not allow lines to extend past 80 characters, use appropriate white space and make sure to use a consistent and attractive indentation scheme.

**Program Submission**

Submit your program code (the **prog3.cpp** file only) at <http://www.mcs.sdsmt.edu/submit>before midnight of the due date. (Your file gets time stamped, so late submissions will be noted and may be given a late penalty!) Be sure to submit to the correct **lecture** section!

**DO YOUR OWN WORK.**