Assignment 7: Data Structures

Assigned: December 01, 2016
Due: December 09, 2016, 11:59:59pm
Note: No late submissions will be accepted for this assignment. This is the absolute deadline in order to leave appropriate time for grading.

Purpose

This assignment contains two exercises based on linked list and stack classes.

Documentation

In addition to the project implementation, you will also need to provide a `README` file. Here you will need to document at least 5 errors you encountered and how you solved them. The errors can be some combination of compile errors, linker errors, runtime, and runtime errors. However, you can not use the same errors you used for the first two assignments. The format needs to be a numbered list where each item describes the error type, the error description, and the steps taken to fix the error. For the description, you can just describe the g++ message for compile/linker errors or the crash / wrong output for runtime errors.

You can also include documentation for places where your implementation is unfinished. If you document buggy or unfinished code and describe what the problem is and and general approach to fixing it, then point deductions will be less severe. This shows me that you are aware of problems in your code and that you at least have an idea of how to fix them. This is completely optional, however, it can only help you as I'm not going to change my test code based on your documentation. That is, if you report an error I would have otherwise missed, you won't lose any points for it.

Exercise 1: Palindrome

Write a program that uses a stack object to determine if a string is a palindrome (i.e. the string is spelled identically backward and forward). The program should ignore spaces and punctuation.

Go ahead and start your program by reading in a C-style string from standard input using the getline function. You may assume a limit of 100 characters on the string. Your algrotihm must make use of a stack (of type char). Use the implementation of Stack from stack.h (don't change the file).

Ignore spacing, punctuation, special characters, and digits (i.e. only count letters). Upper and lower case equivalent letters should be counted as the same (e.g. 'B' and 'b' are matching letters).

Since you are reading data into a C-style string to begin, you may use any of the libraries <iostream>, <cstring>, and <cctype>.

Sample runs (user input is underlined):
Please enter a string:
> ABCDEFGHGFEDCBA

“ABCDEFGHGFEDCBA” IS a palindrome

Please enter a string:
> The quick brown fox

“The quick brown fox” is NOT a palindrome

Please enter a string:
> Cigar? Toss it in a can. It is so tragic.

“Cigar? Toss it in a can. It is so tragic.” IS a palindrome

Exercise 2: List

Modify the List class so that it has two more functions which will allow inserts and removes from anywhere in the linked list. Your functions should be called:

- insertMiddle
- removeMiddle

Your functions should have all the same features as the given insert and remove functions except that yours have one extra parameter. The second parameter on each of your functions should be of type int, representing the position at which to insert/delete.

Sample calls for a list of integers:
L.insertMiddle(345, 5); //attempts to insert the value 345 as the 5th item in the list
L.removeMiddle(x, 10); //attempts to delete the 10th item in the list and captures its value into x.

For insertMiddle, if the position number is larger than the number of items in the list, just insert the item at the back. If it's too small (0 or less), insert at the front.

For removeMiddle, return false if the position is invalid (without removing anything).

You can use menu7.cpp to test these features.

Submitting

- Use the tar utility to archive your palindrome.cpp and list.h files
- Name the tar file according to the scheme: a7_<*last_name*>_<*first_name*>.tar
- Submit the tar file to the appropriate blackboard assignment link