**COMP 2611: Data Structures**

**Assignment #1: Manipulating Stacks, Queues and Linked Lists**

**Date Due: 11:55 p.m., 9th June, 2019**

## Overview

This assignment is designed to give you experience in
(a) manipulating stacks and queues implemented with linked lists, and
b) writing both recursive and non-recursive linked list functions

## Description
## Part (a)

You are supplied with two (2) Dev C++ project containing various `.h` and `.cpp` files. You do not have to modify the project. The code for this assignment must be written in the `Main.cpp` file. There is no need to modify the code in the other files supplied.

## Structure Definition of a Node in a Stack, Queue and Linked List (Defined in NodeType.h)

The functions that have to be written for this assignment assume that a node in a stack, queue and linked list is defined as follows:

```
struct Node {
      char data;
      Node * next;
};
```

Write a program which:

(1) Reads a series of pairs of strings from a text file, `strings.txt`. A string can consist of spaces and other punctuation characters (e.g., `-`, `'`, `.`).

(2) Stores both each pair in separate queues `input` and `output` and determines if `output` is possible through stack permutation
For example:
- if the `input` queue has the string *apple* and the `output` queue contains the string the *lappe* the stack permutation is possible.
- if the `input` queue has the string *hello* and the `output` queue contains the string the *holde* the stack permutation is not possible.

A *stack permutation* is a permutation of objects in the given input queue which is done by transferring elements from input queue to the output queue with the help of a stack.
The well-defined rules are:
- Use inbuilt push, pop functions in the single stack.
- Stack and input queue must be empty at the end.

**COMP 2611: Data Structures**

**Assignment #1: Manipulating Stacks, Queues and Linked Lists**

**Date Due: 11:55 p.m., 9th June, 2019**

**Part (b)**

In addition to the usual linked list operations, write the code for the following functions:

| Return Type | Prototype of Function and Description |
|---|---|
| Node * | **Node * recursiveCopy (Node *top)**<br><br>Creates a copy of the linked list. Returns the address of the first node of the new list. |
| Node * | **Node * removeAll (Node *top, char c)**<br><br>Removes all the occurrences of *c* in the linked list using recursion. Returns the address of the first node of the modified list. A helper method may be used if necessary. |
| Node * | **Node * nonRecRemoveAll (Node *top, char c)**<br><br>Removes all the occurrences of *c* in the linked list without the use of recursion. Returns the address of the first node of the modified list. |
| Node * | **Node * nonRecursiveMerge (Node *top1, Node *top2)**<br><br>Merges two linked lists (assumed to be in sorted order) and produces a new linked list, also in sorted order. Returns the address of the first node of the merged list. |
| char | **int compare (Node * top1, Node * top2)**<br><br>Compares the two linked lists of characters based on "dictionary" ordering. A linked list of characters represents a string. The *compare()* function should behave like the *strcmp()* function in C, except that its parameters are two linked lists rather than two strings. |
| void | **void nonRecursivePrint(Node * top)**<br><br>Prints the contents of the linked list from the end of the linked list without using recursion. |
| Node * | **Node * nthNodeFromEnd(Node* top, int n)**<br><br>Find and returns the nth node of the linked list starting from the end of the linked list. You may use a stack in this implementation |
| Node* | **Node * segregateEvenOdd(Node * top)**<br><br>Modify the linked list such that all the ASCII equivalence of a character that is an even number appear before all the ASCII equivalence of a character that odd numbers in the modified linked list. |

Add code to Main.cpp which tests all the functions in the table above.

NB: In all the functions above, *top* is the address of the first node of a linked list.

**COMP 2611: Data Structures**

**Assignment #1: Manipulating Stacks, Queues and Linked Lists**

**Date Due: 11:55 p.m., 9th June, 2019**

**Files Required (Available at Course Web Site in myElearning)**

| File |
|---|
| NodeType.h |
| Stack.h |
| Stack.cpp |
| Queue.h |
| Queue.cpp |
| LinkedList.h |
| LinkedList.cpp |
| Main.cpp |
| Assignment1.dev |

**Submission Details**

Submit one zipped folder labelled with your UWI student Id number. The contents of the zipped folder are the Dev C++ project files, all the .cpp and .h files and the .txt file.