Homework 4 – Circular Singly Linked List

Topics: This set of problems is based on concepts from **Chapter 3.2-3.4** of your Data Structures & Algorithms textbook and **Chapter 17** of your C++ Early Objects book. Make sure to read through the book chapters and review your lecture notes before you attempt this exercise. Please be sincere in your attempt. You should expect to see similar problems on your exam.

Points: Counts towards your participation grade. **You must make an honest attempt at all problems to get a full participation grade.**

Team-based? Individual

Due: **Section 05 (Mon/Wed) - Monday, 2015-Mar-16 @ 1pm in class.**

Section 01 (Tue/Thu) - Tuesday, 2015-Mar-17 @ 1pm in class.

Late work will not be accepted.

Hand in details: You must turn in your work as a printed or handwritten paper copy. Emailed copies will not be accepted. Please write legibly. Provide your name in your hand in.

Caution: **Cases of plagiarism will be dealt in a strict manner and according to the guidelines described on the course syllabus and university policies on academic dishonesty.**

Getting Started: Problems 1-4 are based on the code in hw4.cpp available for you to download from Titanium.

Problem 1:

Write the destructor for your CLinkedList class. Make sure to delete all nodes. Include here your C++ code for the destructor.

Problem 2:

Modify the CLinkedList class to add a member function for inserting a new node immediately before the cursor. Cursor will point to the first inserted node in case the list was empty to start with.

Prototype: `void insert(int value);`

where value is the info for the new node. Print the empty list, next insert values 10, 20, and 30. Next print the list. Below is one sample run.

```
List is empty.
```

```
Inserting 10

Inserting 20

Inserting 30

List: 10 20 30
```

First give figures (hand drawn) for the above three insertions to your list, show all pointer manipulations. Next include C++ code for your insert and main functions here. Include the aforementioned sample run.

Problem 3:

Modify the CLinkedList class to add a member function for deleting nodes with info within a certain range:

Prototype: `void deleteInRange(int min, int max);`

Deletes all nodes with info equal or higher than the min and equal or lower than the max. Be sure to set the cursor to null in case all nodes get deleted. The function does nothing if no node has info between min and max.

First create a list using insert with nodes having values 20, 10, 80, 60, 30, 90, 15. Next print your list by calling the printList function. Next delete nodes with info in range [10-30]. Call your printList function again. Below is a sample run.

```
List: 20 10 80 60 30 90 15

Nodes deleted in range [10-30]: 20 10 30 15

List: 80 60 90
```

Include C++ code for your deleteInRange and updated main functions here. Include the aforementioned sample run.

Problem 4:

Modify the CLinkedList class to add a member function for splitting the list:

Prototype: `void split(CLinkedList & mylist);`

The member function splits the list into two equal parts if the number of nodes is even. If the number of nodes is odd then the input list will be split unequally such that one of the output lists will contain an extra node. You should create a list in your main function by inserting some number of nodes, let's name it as first. Next you should create an empty list, let's name it as second. Pass your second list to the split function by reference. After the split function executes the second list will contain the second half and the first list will contain the first half of your original list. Look at the below sample run. Also, refer to the commented code in main function of your hw4.cpp for testing your split function.

First create a list using insert with nodes described as in the below two sample runs. Next print your list by calling the printList function. Call your split function then print contents of the two lists.

Sample run one:

```
List: 10 20 30 5 15 4

First List: 10 20 30

Second List: 5 15 4
```

Sample run two:

```
List: 10 20 30 5 15 4 12

First List: 10 20 30 5

Second List: 15 4 12
```

Firstly, give figures for your list (hand drawn) as it looks before and after the split and the intermediate steps needed to split the list, show all pointer manipulations. Secondly, give **pseudo-code** of your split algorithm. Lastly include C++ code for your split and updated main functions here along with two sample runs of your program as shown above. Additionally, be sure to handle the following cases for your input list: the list to split is empty, input list has one node, input list has two nodes.