Langara College
Department of Computing Science & Information Systems
CPSC1160 – Algorithms and Data Structures I
Lab_08: Linked Lists
Dr. Bita Shadgar

1. Instructions

a. Writing your program with nice style is part of your evaluation. Indentation, documentation, modularization and selecting good names for your variables and constants are important.
b. In order to **not to lose mark about indentation**, make sure that find and replace all the tabs in your final program with 4 spaces, then save the code and submit it. This is due to different size of tab in different word processors (editors). It happens sometimes that your program lose right indentation when it is open in a new editor.
c. Read whole assignment first and make sure that you understand different parts of assignment and due dates. If you have any doubt or you are not clear about the assignment, you should ask in lab sessions or office hours. There is no grantee to get answer for your questions in this regard via email out of those times.
d. Create a folder named **Lab8**.
e. Inside folder **Lab8,** create a file for each problem.

Problem 1: [10 marks] Online quiz (Filename: **Quiz.pdf**)

Answer Multiple Choice Quiz at www.cs.armstrong.edu/liang/cpp3e/quiz.html for chapters 20 and 24 (Chapter 24 is optional). Then save your results as an image and paste them into a text file. Finally convert text file to pdf version named Quiz.pdf.

Problem 2: [10 marks] Generate a LinkedList (TestLinkedList.cpp, LinkedList.h)

a. **[5 marks]** Write a function named **genLinkedList** that receives an integer, n, as its parameter, and generates and returns a linkedList of integers with **n** non-negative random elements less than 100, using following function signature.

```
void genLinkedList(int n, LinkedList<int> & list);
```

b. **[5 marks]** Write a function named **printList** that receives a linkedList variable as its parameter and prints out the elements of the list from head to tail.

```
void printList(LinkedList<int> list);
```

Hints: You are not allowed to use built-in libraries of C++ such as slist or list. However, you can use the LinkedList class that has been discussed in the lectures. (Copy and paste needed methods from lectures' slides into **LinkedList.h** file. Then, include **LinkedList.h** to your test file, **TestLinkedList.cpp**). Furthermore, you can add more function to LinkedList class if you need.

Your final code must be compatible to the following function as main function of **TestLinkedList.cpp**.

```
int main(){
     // Use the function from last assignment
     int n = getInput();

     //Test Problem 2
     LinkedList<int> list;
     genLinkedList(n, list);
     cout << "The random list: " ;
     printList(list);

     //Test Problem 3
     LinkedList<int> first;
     LinkedList<int> second;
     split(list, first, second);
     cout << "The list after splitting: " ;
     printList(list);
     cout << "The first half of list: " ;
     printList(first);
     cout << "The second half of list: " ;
     printList(second);

     //Test Problem 5 (explicitly) and Problem 4 (implicitly)
     mergeSort(list);
     cout << "The list after sorting: " ;
     printList(list);

     return 0;
}
```

Problem 3: [12 marks] Split Linked List Class (filenames: **TestLinkedList.cpp)**

Given a list, write a function named **split** that splits the list into two sublists — one for the front half, and one for the back half. The following line represents the function signature:

```
void split(LinkedList<int> source,
           LinkedList<int>& frontList, LinkedList<int>& backList);
```

If the number of elements is odd, the extra element should go in the front list. So split() on the list {2, 3, 5, 7, 11} should yield the two lists {2, 3, 5} and {7, 11}. Getting this right for all the cases is harder than it looks. You should check your solution against a few cases (length = 1, length = 2, length = 3, length=4) to make sure that the list gets split correctly near the short-list boundary conditions. If it works right for length=4, it probably works right for length=1000. You will probably need special case code to deal with the (length <2) cases.

Hint. Probably the simplest strategy is to use the length of the list, then use a `for` loop to hop over the right number of nodes to find the last node of the front half, and then cut the list at that point.

Problem 4: [12 marks] Merge Two Sorted Linked List (filenames: **TestLinkedList.cpp)**

Write a merge() function that takes two linked lists, each of which is sorted in increasing order, and merges the two together into one linked list which is in increasing order.

Problem 5: [6 marks] Merge Sort (filenames: **TestLinkedList.cpp)**

Given split() and merge(), it is pretty easy to write a classic recursive mergeSort(): split the list into two smaller lists, recursively sort those lists, and finally merge the two sorted lists together into a single sorted list. Ironically, this problem is easier than either split() or merge(). Here is the function signature of mergeSort:

```
void mergeSort(LinkedList<int>& list);
```

Problem 6: [**5 marks**] **Bonus Mark** - Calculate time complexity of your functions related to problems 3, 4 and 5. (**answers.pdf**).

Due date

- By the end of the lab time, demonstrate **Problem 2**.
- By 11:59pm on Monday 13, March 2017, submit a zip file named Lab8.zip which includes 4 files named **LinkedList.h**, **TestLinkedList.cpp**, **answers.pdf** and **Quiz.pdf** to D2L.