Computer Science II – Data Structures and Abstraction (CS23001)

Lab 5: Linked List

Objectives:

The goal of this lab is to learn the following:

- Learn about Stack

Lab Submissions:

Please submit one Zip file containing all Source Code organized in separate folders along with evidence of working code, online through Blackboard by the end of the day on next Friday. No email submissions will be accepted. Name the Zip folder *yourname_Labx* (i.e. *AGuercio_Lab2*)

Please ensure that your source code is properly organized. Create separate folders for each exercise and ensure that all needed files (headers, source, makefile) are included in each folder, Use exercise numbers as folder names. Run each exercise to test your code and save a screen shot showing the working code. Zip all folders together with screen shots before submitting it online.

Your code must have proper indentation, appropriate comments, pre and post conditions, and test cases. Test cases that you have used to test your program should be included as a comment on top of your driver file. Be sure you choose the **Test Cases** according to the "Testing" rules covered both in class/slides and in your book. Also be sure that you always separate the class declaration from its implementation and you create a Makefile for its compilation. This requirement MUST be satisfied in **every lab** of this course.

Lab Deliverables:

- Complete working source code of each exercise (where it applies).
- Screenshot(s) of each exercise to show that the code works.

Lab Grading:

The grade for each lab is based on lab attendance and working source code. Lab attendance and submission of Source Code along with Screen shots are mandatory for each lab. Grade distribution includes: Source code (75%), use of proper indentation and commenting of the code (5%), pre-conditions and post-conditions (5%), test cases (5%) and the evidence of working code in the form of screen shots (10%).

Good Luck!

5.1 Linked List Implementation

A linked list is a linear structure consisting of nodes. Each node has a data part that holds information and a reference to the next node. The list is able to grow in size as needed.

There are four main operations associated with stacks; 1) putting things in the linked list which is referred to as **insert**, 2) removing things from the linked list which is referred to as **remove**, 3) printing all the items in the linked list which is referred to as **traverse**, and 4) finding a node that contains a specific data which is referred to as **find**.

Exercise 5.1

Write a program that stores integers in a linked list. Please refer the following data structure for the implementation. Your program will ask users to input an integer and then insert the integer at the front of the list. Call your application *ex51.cpp*. Please do not test remove yet. You will need the method for 5.2.

```
class ListNode{
public:
    int data;
    ListNode *next;
};
class linkedList{
private:
    ListNode *head;
public:
    linkedList(){ head = NULL;}
    void insertFront(int data);
    void remove(ListNode *prev, ListNode *curr);
    void traverse();
    ListNode* find(int target);
};
```

5.2 Additional Functions

We have discussed few more FIND methods in the class. The find method can be improved in different forms, such as Find and Replace, Find and Insert, and Find and Remove.

Exercise 5.2

Write and test the additional methods, findAndReplace, findAndInsert, and findAndRemove. Call your application *ex52.cpp*.

- findAndReplace: Will replace an item to the target value.
- findAndInsert: Will insert an item next to the target value.
- findAndRemove: Will remove an item from the list.

Congratulations!

You have just completed another Lab!