**Lab Assignment #1 – Using Fundamental Data Structures**

Due Date: By the dropbox deadline.

Purpose: The purpose of this Lab assignment is to:

1. Design and develop Applications that incorporate

* Linked Lists

References: Read the course’s text chapter 3 and the lecture slides. This material provides the necessary information that you need to complete the exercises.

**Instructions:**

You **MUST** create a short demo video of your solution. Do not show yourself in the video. Upload your video in your personal youtube account or google-drive account and share it with the instructor **only**. Do not share it publicly. During submission at the dropbox, **write the link of your video** in the **Comments** **box** (present near the bottom of the submission page). Next, create a zip file of your solution as mentioned below in section **Submission Rules**, upload that zip file, and submit.

You **must** name a relevant Eclipse project according to the following rule:

YourFirstname\_YourLastname\_COMP254\_Labnumber\_ExerciseNumber.

Example: If student name is John Smith, the name of Eclipse project for Ex1 of Lab1 should be **John\_Smith\_COMP254\_Lab1\_Ex1**

**Submission Rules:**

Compress all your Eclipse projects as a **single** **zip** filethat is named according to the following rule: YourFirstname\_YourLastname\_COMP254\_Labnumber.zip

Example: **John\_Smith\_COMP254\_Lab1.zip**

Submit the above single zip file using the procedure mentioned in section **Instructions** above.

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality:**   * Correct implementation of requirements * Code explanation when asked * Answer questions on the topic when asked | 90% |
| **Object-Oriented design**: correct design of classes and methods similarly to chapter 3 examples. | 10% |
| **Total** | 100% |

**Exercise 1**

In this exercise, you will use the *DoublyLinkedList* implementation of the textbook (week 2 lecture examples). Write a non-static method named *concatenate*(***DoublyLinkedList***secondList) for concatenating two doubly linked lists, say firstList and secondList. You should pass the

secondList as a parameter to the method. You should invoke the method on the firstList. Remember that a doubly linked list always has a header sentinel node and a trailer sentinel node. Connect the end of firstList to the beginning of secondList. Make sure that the resultant doubly linked list has only one header sentinel node and one trailer sentinel node.

Write a main method to test the new method.

(5 marks)

**Exercise 2**

In this exercise, you will add a method ***swapNodes***to *SinglyLinkedList* class from week 2 lecture examples. This method should swap two nodes *node1* and *node2* (and not just their contents) given references only to *node1* and *node2*. The new method should check if *node1* and *node2* are the same node, etc. Write the main method to test the *swapNodes* method. **Hint**: You may need to traverse the list.

(5 marks)