Lab Assignment #4 – Using ADT Stacks, Queues, and Lists

Due Date: Friday, Week 8

Purpose: The purpose of this Lab assignment is to:

- Design algorithms that describe operations on ADT stacks, queues, and lists
- Implement and test appropriate methods in Java or Python

References: Read the course's text chapter 6, 7 and the lecture slides. This material provides the necessary information that you need to complete the exercises.

Be sure to read the following general instructions carefully:

- This assignment must be completed individually by all the students.
- You will have to **demonstrate your solution in a scheduled lab session** and upload the solution on **eCentennial** through the assignment link.

Exercise 1

Suppose we want to extend the **PositionalList** ADT with a method, ***indexOf(p)***, that returns the current index of the element stored at position p. Write this method using only other methods of the **PositionalList** interface (not details of our LinkedPositionalList implementation). Write the necessary code to test the method. **Hint:** Count the steps while traversing the list until encountering position p.

(5 marks)

Exercise 2

Implement a method with signature *transfer(S, T)* that transfers all elements from stack S onto stack T, so that the element that starts at the top of S is the first to be inserted onto T, and the element at the bottom of S ends up at the top of T. Write the necessary code to test the method.

(2 marks)

Exercise 3

Implement a method with signature ***concatenate(LinkedQueue<E> Q2)*** for the **LinkedQueue<E>** class that takes all elements of Q2 and appends them to the end of the original queue. The operation should run in **O(1)** time and should result in Q2 being an empty queue. Write the necessary code to test the method. **Hint:** You may just modify the SinglyLinkedList class to add necessary support.

(3 marks)

Evaluation:

Correct implementation of requirements: • Correct ADT data structure algorithm • Correct Java or Python implementation • Explanation of algorithm when asked	90%
Friendly I/O	10%
Total	100%

You must name your Eclipse project according to the following rule:
YourFullname_COMP254Labnumber.

Example: **JohnSmith_ COMP254Lab4**

Create a package for each exercise in your project.

Submission rules:

Submit your modules as **zip files** that are named according to the following rule:
YourFullname_ COMP254Labnumber.zip

Example: **JohnSmith_ COMP254Lab4.zip**

Use 7-zip to compress files (https://www.7-zip.org/download.html). DO NOT use RAR or other software.