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

**Due Date:** Midnight of March 6, 2019 (Sunday)

**Purpose:** The purpose of this assignment is to help you:

1. Understand operations of ADT stacks, queues

**Instructions**: Be sure to read the following general instructions carefully:

This lab should be completed individually by all the students. You **have to demonstrate** **your solution during the lab**, and submit your solution **through the dropbox**. Name your submission according to the following rule: **studentID(yourlastname)\_Labnumber.zip**. e.g., 300123456(**smith)\_Lab#4**.zip

**Rubric**

|  |  |
| --- | --- |
| **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% |

**Exercise 1 [3 marks]**

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.

**Exercise 2 [2 marks]**

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.

**Exercise 3 [5 marks]**

An array is sparse if most of its entries are null. A list L can be used to implement such an array, A, efficiently. In particular, for each non null cell A[i], we can store a pair (i,e) in L, where e is the element stored at A[i]. This approach allows us to represent A using O(m) storage, where m is the number of non null entries in A.

Describe and analyze efficient ways of performing the methods of the array list ADT on such a representation. **Hint**: There is a trade-off between insertion and searching depending on whether the entries in L are sorted.