University of Windsor

School of Computer Science

60-254 Data Structures and Algorithms – Winter 2017
Lab Assignment 2

Deadline: **February 6/8, 2017** (15 minutes prior to the end of the lab)

Regulations: This assignment must be done individually. Any similarity between your code/answers and another student's, or a carbon-copy of an answer found in the web will be considered plagiarism. None of the built-in classes/methods/functions of Java/C/C++ or any other language can be used for the ADTs. You must implement **your own** ADTs and their operations.

Objective: The aim of this assignment is to obtain hands on experience in implementing ADTs, and understanding the theoretical concepts seen in class on algorithm design and analysis. At the end of this assignment, students will know how to implement Stacks, Queues and Linked Lists and how to analyze the complexity of their operations.

Tasks:

1. Using your favorite programming language (Java is suggested), implement a singly linked list. Implement the following operations: addFirst, removeFirst, addLast, removeLast, getFirst, getLast, size.
2. Implement a stack on the singly linked list with the operations of Lab Assignment 1. Hint: Using the same Stack class you implemented, change the array to an object of the singly linked list class. The functionality of push and pop is based on the methods of the linked list class.
3. Practical application 1: Implement the balanced-bracket checker algorithm of Lab Assignment 1 using the singly linked list implementation of the stack.
4. Implement a queue on the singly linked list of Item #1, with the following operations: enqueue, dequeue, front, size, isEmpty.
5. Practical application 2: Using the queue of Item #4, implement a printer queue. For simplicity, the jobs to be sent to the printer are numbers (integers with any number of digits). These numbers have to be entered from the standard input (e.g., the keyboard) and stored in the queue. Once all numbers have been entered, they should be dequeued from the queue and printed out (shown on the screen) in the same order they were entered – that is, first-in-first-out. Note: when all the numbers have been printed out, the queue must be empty, and so your program should show a message that says “the queue is empty”.
6. What is the worst-case time complexity of the printer queue algorithm to print n jobs? Why?

Submission:

1. Your assignment must be submitted during the lab session in the section you are registered in. Any submission *on or 15 minutes* prior to the end of the lab session will **not** be accepted and a **zero** mark will be given. Late assignments will be given a **zero** mark. Submissions by email will **not** be accepted.
2. Provide the source code for all your programs.
3. Run the balanced-bracket checker during the lab and show they work for various sample inputs, including (more may be included during the evaluation in the lab – a and d are balanced):
    a. (9*[3*{[(3+3)/5]*7}])
    b. {3*(2+[3-[4/[6/9]]]})
    c. ((3*(9-(4*(6-5))))
    d. {2-{3*{6/[[[(((9-0)))]]]}}/7}
4. Run the printer queue with the following inputs:
    a. 120 30 60 59 1554 9666 8444
    b. 23 555 1000000 2 19
5. Explain how your singly linked list, stack and queue work. These will be asked when the lab assignment is submitted. Marks will be deducted if not clear how things work.