DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
CONCORDIA UNIVERSITY
COMP 352: Data Structure and Algorithms
Winter 2018
ASSIGNMENT 3
Due date and time: Monday March 19th by 11:59 pm

***Written Questions (50 marks)**:*

Q1. Answer the following questions:

a) Let T be an ordered tree (i.e., the children of each node are ordered) with more than one node. Is it possible that the in-order traversal of T visits the nodes in the same order as the pre-order traversal? If so, give an example; otherwise argue why it cannot happen. Likewise, is it possible that post-order traversal of T visits the nodes in the reverse order as pre-order traversal? If so, give an example; otherwise argue why it cannot happen.
b) Answer the previous questions if T is a proper binary tree with more than one node.

Q2. Answer the following questions:

a) Show (diagrammatically) all the steps involved in the heap sort on the following input sequence: 7, 45, 1, 21, 2, 64, 4, 18, 9, 6. You should sort the sequence in the increasing order.
b) Repeat part a) if the heap sort is performed **in place**.

Q3. Answer the following questions:

a) Bill suggested that he can implement a *queue* ADT using a *priority queue* and a constant amount of additional space. Recall that a queue supports *enqueue* and *dequeue* operations, and a priority queue supports *min(), removeMin* and *insert* operations. Explain how Bill can achieve this by showing the necessary pseudo code for implementing the queue ADT.
b) Referring to a) above, what are the tight big-Oh time complexities of the *enqueue* and *dequeue* operations of the newly designed queue given that the priority queue is implemented using a heap? Explain your answer.

***Programming Questions (50 marks)**:*

Q4. In class, we discussed the priority queue (PQ) ADT implemented using min-heap. In a min-heap, the element of the heap with the smallest key is the root of the binary tree. On the other hand, a max-heap has as root the element with the biggest key, and the relationship between the keys of a node and its parent is reversed of that of a min-heap. We also discussed an array-based implementation of heaps.

In this assignment, your task is to implement your own **flexible priority queue ADT** using both min- and max-heap in Java. The specifications of the flexible priority queue are provided in the following. The heap(s) must be implemented from scratch using an array that is automatically extendable. You are not allowed to use any list (including arraylist), tree, vector, or heap implementation already available in Java. You must not duplicate code for implementing min- and max-heaps (i.e. the same code must be used for constructing min or max heaps. Hence think of a flexible way to parameterize your code for either min- or max heap). The following are the access methods of the flexible priority queue ADT:

remove(): removes and returns the entry (a key, value pair) with the smallest or biggest key depending on the current state of the priority queue (either Min or Max).

insert (k,v): Insert the (k,v) entry which is a key(k), value(v) pair to the priority queue.

top(): returns the top entry (with the minimum or the maximum key depending on whether it is a Min- or Max-priority queue, without removing the entry.

toggle() transforms a min- to a max-priority queue or vice versa.

switchToMin(): transforms a max- to a min-priority queue; else leave unchanged.

switchToMax(): transforms a min- to a max-priority queue; else leave unchanged.

state (): returns the current state (Min or Max) of the priority queue.

isEmpty(): returns true if the priority queue is empty.

size(): returns the current number of entries in the priority queue

You have to submit the following deliverables:

a) Pseudo code of your implementation of the flexible priority queue ADT using a parameterized heap that is implemented using extendable array. Note that Java code will not be considered as pseudo code. Your pseudo code must be on a higher and more abstract level.
b) Tight big-Oh time complexities of toggle(), switchToMin(), and switchToMax() operations, together with your explanations.
c) Well documented Java source code with 20 different but representative examples demonstrating the functionality of your implemented ADT. These examples should demonstrate all cases of your ADT functionality (e.g., all operations of your ADT, several cases requiring automatic array extension, sufficient number of down and up-heap operations).

Submit all your answers to written questions in PDF or text formats only. For the Java programs, you must submit the source files. The solutions to all the questions should be zipped together into one .zip file and submitted via EAS (Refer to the course outline for more details on submission guidelines).