Data Structures & Applications
Fall 2020
Lab 08 – Priority Queues

Instructor: Saif Hassan **Date: 22nd November, 2020**

Instructions:

- At the end of this Lab, you will have to submit all files on LMS.
- Attempt all the tasks in any environment and submit code on LMS as well.
- File format should be **.zip/.rar** file containing required **.java** files and additional if required.
- File Name should be your **CMSID_Name_Lab08_Section.zip**.
- Create a project named **lab08_dsa** and perform following tasks.
- **.java** files should be as following:
    - **PriorityQueueUsingArray.java** --> contains complete code for Complete **Priority Queue Using Array** Class with implemented required functions.
    - **CompleteBinaryTree.java** --> contains complete code for Complete **Binary Tree** Class with implemented required functions

Note: Labs submission without following above instructions will not be checked. (No any excuse will be entertained.)

Note: Keep this complete lab code with you till the course ends.

Task 01: (Priority Queue using unsorted array)

A **PriorityQueue** is used when the objects are supposed to be processed based on the priority. It is known that a queue follows First-In-First-Out algorithm, but sometimes the elements of the queue are needed to be processed according to the priority, that's when the **PriorityQueue** comes into play. The **PriorityQueue** is based on the priority heap. The elements of the priority queue are ordered according to the natural ordering, or by a Comparator provided at queue construction time, depending on which constructor is used.

You have been provided file named **"PriorityQueueUsingArray.java"**. Your task is to complete **methods** within **PriorityQueueUsingArray** class.

In this case, running time for insert will be $O(1)$ and for extractMax/getMax will be $O(n)$.

Task 02: (Priority Queue using sorted array)

After completing **Task 01,** your task is to modify task 01 to implement priority queue using sorted array (as discussed in lecture 8.1). You just have to modify **insert** method to add element in ascending order and **extractMax** and **getMax** method to return last element.

In this case, running time for insert will be $O(n)$ and for extractMax/getMax will be $O(1)$.

Task 03: (Complete Binary Tree using array)

A **complete binary tree** is a **binary tree** in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

Write a code from scratch for implementation of complete binary tree. Create and Design required attributes methods (as discussed in lecture 8.3).

Test your code by calling insert, remove, changing value, searching methods. Each time binary tree muse be complete even after calling any of above methods.