COMP 203 DATA STRUCTURES AND ALGORITHMS

HOMEWORK 4 (Total=100 points)

Deadline: 29.12.2024 23:59

Read the questions and rules carefully. They are clear and well defined.

Rules:

1. No Cheating: You are not allowed to collaborate with your friends and use any kind of websites or AI. If your homework gives a sign of any of them, **directly it will be graded as zero**.

2. Goal: Please do your homework alone. Our main aim is to **learn** whatever we cover so far.

3. Submission: Submit your homework in **2 java files. No other file types will be accepted. You will submit only 2 java files. DON'T USE ZIP/RAR etc. In these cases, your points will be deducted by 30%.**

4. Coding policy: Explain your code in comments. **This is a must!**

5. Latency policy: A 10% deduction will be applied for each day of late submission.

QUESTIONS

1. Implement Priority Queue abstract data structure using fixed size array data structure. Use the sorted implementation approach. Implement the following classes and methods: **(50pt)**

Example: 4,5,6,7,8,9. In this PQ 4 has the highest priority.

1. Implement FixedSizePriorityQueue<T> class with constructor with int capacity variable. **(5pt)**

   Hint: FixedSizePriorityQueue has:

   private Object[] array;

   private int size;

   private int capacity.
2. add(E addedValue) that adds the given value *addedValue* to PQ. *addedValue is used as a priority value.* **(10pt)**

   Hint: First, find the best spot in the PQ to insert the *addedValue.*
3. T removeMin() that removes the highest priority in PQ and returns this value.**(10pt)**

   Hint: Delete the highest priority element in the queue. Then, shift the elements one backwards not to leave any empty spot in array.

   For example;

Before: 4,5,6,7,8,9.

After dequeue: 5,6,7,8,9.

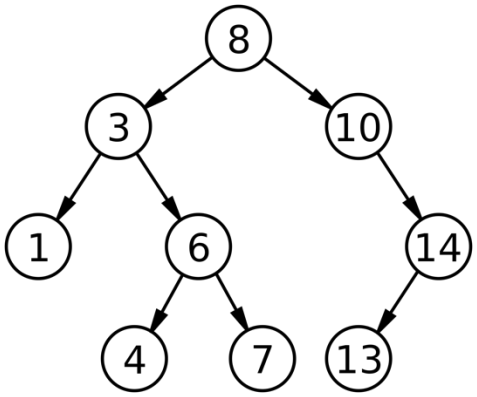
4. size() that returns the number of elements in the PQ. **(5pt)**
5. printPriorityQueue() that prints the PQ. **(5pt)**
6. Test your methods in the main by creating the following PQ with the integer data type. (Creating the given PQ: **5 pt**, testing 4 methods: **10 pt**)
   PQ: 4,5,6,7,8,9.

Include comments of your code for each method and class.

Submit FixedSizePriorityQueue.java to Canvas.

2. Write a java program that checks if the given binary tree is a heap or not. Implement Binary Tree abstract data structure using Node data structure. Implement the following classes and methods: **(50pt)**

1. Implement Node class with (Integer data type) for Binary tree and its constructor with the parameter int data. **(5pt)**
2. Implement BinaryTree class with the class variable and empty constructor. **(5pt)**
3. Boolean isHeap(Node root) that checks if the given binary tree is a heap or not. **(25pt)**
4. Test your method in the main by creating the following binary tree. (Creating the given binary tree:**10 pt**, testing your isHeap(Node root) method:**5 pt**)



Include comments of your code for each method and class.

Submit BinaryTree.java to Canvas.