**Lab Assignment #5 – Using Trees and Priority Queues**

Due Date: Friday, Week 10

Purpose: The purpose of this Lab assignment is to:

1. Design algorithms that describe operations on ADT Trees and priority queues.
2. Implement and test appropriate methods in Java or Python

References: Read the course’s text chapter 8, 9 and the lecture slides. This material provides the necessary information that you need to complete the exercises.

Be sure to read the following general instructions carefully:

- This assignment must be completed individually by all the students.

- See the naming and submission rules at the end of this document

**Exercise 1**

**If your first name starts with a letter from A-J inclusively:**

Design the algorithm and method **following operations** for a binary tree T:

* preorderNext(p): Return the position visited after p in a preorder traversal of T (or null if p is the last node visited).
* inorderNext(p): Return the position visited after p in an inorder traversal of T (or null if p is the last node visited).

Write a Java/Python to test your solution.

What are the worst-case running times of your algorithms?

**If your first name starts with a letter from K-Z inclusively:**

1. Design the algorithm and method **following operations** for a binary tree T:

* inorderNext(p): Return the position visited after p in an inorder traversal of T (or null if p is the last node visited).
* postorderNext(p): Return the position visited after p in a postorder traversal of T (or null if p is the last node visited).

Write a Java/Python to test your solution.

What are the worst-case running times of your algorithms?

(5 marks)

**Exercise 2**

Give an efficient algorithm that computes and prints, for every position p of a tree T, the element of p followed by the height of p’s subtree. Write a Java/Python to test your solution.

**Hint**: Use a postorder traversal to find the height of each subtree. The height for a subtree at p will be 0 if p is a leaf and otherwise one more than the height of the max child. Print out the element at p and its computed height during the postorder visit.

(3 marks)

**Exercise 3**

**If your first name starts with a letter from A-J inclusively:**

Give an alternative implementation of the HeapPriorityQueue’s upheap method that uses recursion (and no loop). **Hint**: Do a single upward swap and recur (if necessary).

**If your first name starts with a letter from K-Z inclusively:**

Reimplement the SortedPriorityQueue using java array. Make sure to maintain removeMin’s O(1) performance.

(2 marks)

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality:**   * Correct implementation of requirements * Code demonstration and brief explanation in a short video | 70%  10% |
| **Object-Oriented design**:   * Correct design of classes and methods similarly to chapter 3 examples. * Correct use of generics * Correct use of naming guidelines for classes, variables, methods. | 15%  5% |
| **Total** | 100% |

**Naming and Submission Rules:**

You must **name your Eclipse project** according to the following rule:

**YourFullname\_COMP254Labnumber**. Example: **JohnSmith\_COMP254Lab1**

You must name package names **com.exercisenumber.yourfirstname.yourlastname**, for example: com.exercise1.john.smith

Provide your **student number and full name as a comment** at the top of main method for each exercise.

**Archive your project in a zip file** named according to the following rule:

**YourFullname\_COMP254Labnumber.zip**

Example: **JohnSmith\_COMP254Lab1.zip**

Upload the zip file on eCentennial using the Assignment link.