**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.

- You will have to **demonstrate your solution in a scheduled lab session** and upload the solution on **eCentennial** through the assignment link.

**Exercise 1**

Design the algorithm and method **for one** of the 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).
* 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**

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).

(2 marks)

**Evaluation:**

|  |  |
| --- | --- |
| **Correct implementation of requirements:**   * Correct ADT data structure algorithm * Correct Java or Python implementation * Explanation of algorithm when asked | 90% |
| **Friendly I/O** | 10% |
| **Total** | 100% |

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

**YourFullname\_COMP254Labnumber\_Exercisenumber**.

Example: **JohnSmith\_ COMP254Lab4\_Ex1**

**Submission rules:**

Submit your modules as **zip files** that are named according to the following rule:

**YourFullname\_ COMP254Labnumber\_Exercisenumber.zip**

Example: **JohnSmith\_ COMP254Lab4\_Ex1.zip**

Use 7-zip to compress files (https://www.7-zip.org/download.html).