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

Due Date: By the dropbox deadline.

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

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.

**Instructions:**

You **MUST** create a short demo video of your solution. Do not show yourself in the video. Upload your video in your personal youtube account or google-drive account and share it with the instructor **only**. Do not share it publicly. During submission at the dropbox, **write the link of your video** in the **Comments** **box** (present near the bottom of the submission page). Next, create a zip file of your solution as mentioned below in section **Submission Rules**, upload that zip file, and submit.

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

YourFirstname\_YourLastname\_COMP254\_Labnumber\_ExerciseNumber.

Example: If student name is John Smith, the name of Eclipse project for Ex1 of Lab1 should be **John\_Smith\_COMP254\_Lab1\_Ex1**

**Submission Rules:**

Compress all your Eclipse projects as a **single** **zip** filethat is named according to the following rule: YourFirstname\_YourLastname\_COMP254\_Labnumber.zip

Example: **John\_Smith\_COMP254\_Lab1.zip**

Submit the above single zip file using the procedure mentioned in section **Instructions** above.

**Evaluation:**

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

**Exercise 1**

Suppose we want to extend the **PositionalList** ADT with a method, **indexOf( *p* )**, that returns the current index of the element stored at position ***p***. Write the body of this method in the existing class **LinkedPositionalList**. Test this method in the **main** method of **LinkedPositionalList**. The class **LinkedPositionalList** is in **Lesson7Examples** posted in the **eCentennial** module “**Lesson Examples (from textbook)**”. **Hint:** Count the steps while traversing the list until encountering position p.

(5 marks)

**Exercise 2**

Provide a method named **printPositionAndHeight** that computes and prints, for every position ***p*** of a tree T, the element of ***p*** followed by the height of the subtree rooted at ***p***. Write this method in the existing class **LinkedBinaryTree**. Test this method in the **main** method of **LinkedBinaryTree**. The class **LinkedBinaryTree** is in **Lesson8Examples** posted in the **eCentennial** module “**Lesson Examples (from textbook)**”. **Hint**: Use a postorder traversal to collect the positions of the tree. Next, print the element of ***p*** followed by the height of the subtree rooted at ***p***.

(5 marks)