**Lab Assignment #7 – Search Trees and Sorting**

Due Date: Friday, Week 13

Purpose: The purpose of this Lab assignment is to:

1. Design algorithms that describe operations on Binary Search trees
2. Design algorithms that implement Sorting operation
3. Implement and test appropriate methods in Java or Python

References: Read the course’s text chapter 11, 12 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

- You will have to provide a **demonstration video for your solution** and upload the video together with the solution on **eCentennial** through the assignment link. See the **video recording instructions** at the end of this document.

**Exercise 1**

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

Our implementation of the *treeSearch* utility, from Code Fragment 11.3, relies on recursion. For a large unbalanced tree, it is possible that Java’s call stack will reach its limit due to the recursive depth. Give an alternative implementation of that method that does not rely on the use of recursion.

**Hint** Use a **loop** to express the repetition.

Write a Java/Python application to test your solution with tree of fig 11.1 from textbook .

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

Our implementation of the *treeSearch* utility, from Code Fragment 11.3, relies on recursion. For a large unbalanced tree, it is possible that Java’s call stack will reach its limit due to the recursive depth. Give an alternative implementation of that method that does not rely on the use of recursion.

**Hint** Use a **loop** to express the repetition.

Write a Java/Python application to test your solution with tree of fig 11.7 from textbook .

(5 marks)

**Exercise 2**

Implement a bottom-up merge-sort for a collection of items by placing each item in its own queue, and then repeatedly merging pairs of queues until all items are sorted in ascending order within a single queue. **Hint:** A **queue of queues** can be very helpful.

Write a Java/Python application to test your solution.

(5 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.