COMP 2230 - Data Structure, Algorithm Analysis, and Program Design

Laboratory No. 5

Marks: 40

Due date and time for submission: Sunday, November 3, 2019.

Method of submission: Moodle.tru.ca

In this lab, we will do more experiments with Binary Tree and get familiar with binary tree operations. You should use the "BinTree.java" and "BinSearchTree.java" code that we have created in the class for this lab.

Q1. [5 Marks] Write a public method "printRootToLeafPath()" that prints out ALL its root-to-leaf paths. Note: you should use internal recursive private method to print all paths.

Q2. [5 Marks] Write a public method "getSum()" that returns the sum of all elements in binary tree. *[Please check that the data are integers, otherwise throws exception]*. Note: you should use internal recursive private method to print all paths.

Q3. [5 Marks] Given two binary tree, can you determine if they are identical or not? Write a program for that.

`Algorithm:`

a) `If both trees are NULL then return true`
b) `If both trees are not NULL then, return TRUE if they are identical.`

Q4. [5 Marks] Write a program for converting a binary tree to its mirror. Mirror of a three is another tree with left and right children of all internal nodes are interchanged. The tree below are the mirrors to each other.

Q5. [5 Marks] In the binary search tree class, write a method called findMax() and findMin() methods that return the largest and smallest keys respectively in the binary search tree. Test the methods.

Q6. [5 Marks] In the binary search tree class implement the search algorithm in a non-recursive manner. Test the method.

Q7. [10 Marks] Write a method that takes two binary search trees as input parameters and returns a third binary search tree that is a merger of the two trees. Test it using a driver (main) program. Note: A naïve method is simple to write. However, a smarter algorithm with less complexity will receive bonus points.

Note:

Make sure, you have exception handling code in place for each exceptional condition. Also, make sure you have tested all the codes with various boundary conditions.

Please submit all the .java files and scree captures of test run of the code. Snapshot of the test run for various data set is very important, please submit that with your source code.

Grading Rubric

1. *Code compiled without any error*	*20%*
2. *Good Comments*	*20%*
3. *All requirements are met*	*40%*
4. *Screenshots of several test runs are included*	*10%*
5. *Apply OOP principals* • *Modularity* • *Good use of classes* • *Encapsulation*	*10%*

Note: All problems will be graded based on the above-mentioned rubric.