CSCI 2110 Data Structures and Algorithms

Laboratory No. 7

Week of November 13th, 2017

In this lab you will implement methods for binary trees. Most of the methods are simple recursive methods.

Marking Scheme

Each exercise carries 10 points. Your final score will be scaled down to a value out of 10.
Working code, Outputs included, Efficient, Good basic comments included: 10/10
No comments: subtract one point
Unnecessarily inefficient: subtract one point
No outputs: subtract two points
Code not working: subtract up to six points depending upon how many methods are incorrect.

***Error checking**: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input.*

***Submission**: All submissions are through Brightspace. Log on dal.ca/brightspace using your Dal NetId. Submissions are pretty straightforward. Instructions will be also be given in the first lab.*
Deadline for submission: Sunday, November 19th, 2017 at 11.55 p.m. (five minutes before midnight).

You will need the following files:
BinaryTree.java, BinarySearchTree.java.
Download these files (alongside the Lab7 link).

This lab requires you to add/modify methods in the BinarySearchTree class. Test all the methods in a client program (BinarySearchTreeDemo.java). Do not hardcode any of the input values in your source code. Prompt the user to enter the values.

Exercise 1: In the binary search tree class, write a method called findMax() that returns the largest key in the binary search tree. Test the method in BinarySearchTreeDemo.java.

```
public T findMax(){…}
```

Note: The largest key is the rightmost node.

Exercise 2: In the binary search tree class, write a method called findMin() that returns the smallest key in the binary search tree. Test the method in BinarySearchTreeDemo.java.

```
public T findMin(){…}
```

Note: The smallest key is the leftmost node.

Exercise 3: The binary search tree class implements the search algorithm in a non-recursive manner. Implement a recursive search algorithm. Test the method in BinarySearchTreeDemo.java. Here's the outline of the solution:

```
//driver method
public BinaryTree<T> recursiveSearch(T key)
{
      if (tree.isEmpty())
            return null;
      else
            return recursiveSearch(tree, key);
}
```

```
//recursive search method
public BinaryTree<T> recursiveSearch(BinaryTree<T> t, T key)
{
      //complete the code
}
```

Exercise 4:
Given a binary tree, write a method that determines whether the tree is a binary search tree. Test it using a driver (main) program.

```
public boolean isBinarySearchTree(BinaryTree<T> t)
```