CSCI 2110 Data Structures and Algorithms
Laboratory No. 9
Release Date: Nov 15th
Due Date: Nov 22nd
23h55 (five minutes to midnight)

Binary Search Trees

The objective of this lab is to help you get familiar with binary search trees (BSTs). Download the example code/files provided along with this lab document. You will need the following files to complete your work:

BinaryTree.java (Generic BinaryTree Class)
BinarySearchTree.java (Generic BinarySearchTree Class)
Exercise1.java (Starter code)

Marking Scheme

Each exercise carries 10 points.
Working code, Outputs included, Efficient, Good basic comments included: 10/10
No comments or poor commenting: subtract one point
Unnecessarily inefficient: subtract one point
No outputs and/or not all test cases covered: subtract up to two points
Code not working: subtract up to six points depending upon how many methods are incorrect. Markers can test cases beyond those shown in sample inputs, and may test methods independently.

Your final score will be scaled down to a value out of 10. For example, if there are three exercises and you score 9/10, 10/10 and 8/10 on the three exercises, your total is 27/30 or 9/10.

Error checking: Unless otherwise specified, you may assume that a 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 to dal.ca/brightspace using your Dal NetId.

What to submit:
Submit one ZIP file containing all source code (files with .java suffixes) and text documents containing sample outputs. For each exercise you will minimally have to submit a class containing your completed methods, a demo class, and sample outputs

Java Versions: Please ensure your code runs with Java 11.

Your final submission should include the following files: *BinaryTree.java, BinarySearchTree.java, Exercise1.java, Exercise1out.txt*

You MUST SUBMIT .java files that are readable. If you submit files that are unreadable such as .class, you will lose points. Please additionally comment out package specifiers.

Exercise 0: Binary Search Tree Methods (no marks)

You have been given the file BinarySearchTree.java. Complete the following methods below:

1. Complete the *findMax* method in the BinarySearchTree.java file. This method should return the maximum data value stored in the binary search tree (i.e. the largest integer or double, the String which is lexicographically last).

2. Complete the *findMin* method in the BinarySearchTree.java file. This method should return the minimum data value stored in the binary search tree (i.e. the smallest integer or double, the String which is lexicographically first).

Test your methods before proceeding:

- Create a simple BST
- Use the *insert* method to build the tree
- Call the *findMax* and *findMin* methods

Exercise 1: Recursive Search

Before you begin, examine the static method *isBinarySearchTree* within the Exercise1.java file. You will use this method to test if your constructed tree is a binary search tree.

Complete the *recursiveSearch* method in the BinarySearchTree.java file. This method returns a BinaryTree whose data value matches the search key. Your solution must be recursive.

- Before you begin coding, do a trace with pen and paper to understand how to recurse through the tree.
- There are two recrusiveSearch methods in the starter code.
    - One to be completed
    - One which acts as a helper method by calling the second

Once your search method has been implemented and tested, you will complete the Exercise1.java code:

- Create a binary search tree which stores integers from user input. You will read in integer values from the user until 0 is entered.
- Construct the BST. The *insert* method is useful here.

Once your BST has been constructed, perform the following operations:

- Print the max element
- Print the min element
- Prompt the user to search for an element in the tree:
    - Search for an element that exists and print the result
    - Search for an element in the tree that doesn't exist and print the result
- Call the *isBinarySearchTree* static method on your tree to ensure it is a BST. Print the result.

Student Generated Test Cases:

- Use the below test case to prove that your BST has been constructed and that the search function works.
- Additionally, perform a second test case where the search function fails. Copy your output to a file named *Exercise1out.txt.*

Input/output

```
Enter int or '0': 1
Enter int or '0': 2
Enter int or '0': 3
Enter int or '0': 4
Enter int or '0': 5
Enter int or '0': 6
Enter int or '0': 7
Enter int or '0': 8
Enter int or '0': 9
Enter int or '0': 0

The max data value in the BST is: 9
The min data value in the BST is: 1

What key would you like to search for?
[INT KEY]>>> 3
Found!


And is it a true BST... Yes!
```