# Course: CSC 115 (J)

## Lab 08: Binary Search Tree

### Objectives

You will be working on a binary search tree class. Before beginning, make sure you are familiar with the concepts presented in Lab 7 related to basic binary trees.

During lab 8, you will:

- Complete the *insert* method of a binary search tree.
- Implement a *level order traversal* of the binary search tree.
- Come to a better understanding of the structure and function of a binary search tree.

### Part I – Inserting into a Binary Search Tree

1. Download *TreeNode.java*, and *DrawableBSTree.java* from connex.
2. Download and complete *BinarySearchTree.java* so that items can be inserted into the binary search tree.
    - The class already includes basic methods, along with a *search* method. Spend a few minutes looking over *search* and make sure you understand how it works. Implementing insert will require you to take a very similar approach with a few important differences.
        - Note in particular how the *compareTo* method is used when comparing two values. *a.compareTo(b)* will return an integer $< 0$ if $a < b$, or $> 0$ if $a > b$. We use this method for comparisons to make our *BinarySearchTree* more flexible. This way it can handle any type of data that implements the Comparable interface.
    - Your task is to complete the *insert* and *recursiveInsert* methods so that data is added to the right part of the binary search tree. Use the comments in the code to help guide your implementation.
3. **Compile** and **run (test)** your code before moving on to the next part. The code in the main will create a binary search tree and add some numbers to it. Make sure they are added in the right place! You can check this visually or by examining the InOrder traversal, which should list the data in ascending order.

### Part II – LevelOrder Traversal

1. Download *Queue.java* from connex. It is the same code given as part of Assignment 4, and will be needed to implement the LevelOrder traversal.
2. Complete the *traversalLevelOrder()* method in *BinarySearchTree.java* so that it returns all the data in the binary search tree as a single String according to a level order traversal.

- A LevelOrder traversal grabs the data from the tree in level order going top down, left to right. It can be implemented using the following algorithm:
    - `Create an empty queue and enqueue the root of the tree.`
    - `while the queue isn't empty,`
        - `dequeue the front item`
        - `add its data to the String`
        - `enqueue its children if they are not null`

3. Make sure to **test** your traversal and compare it to the tree being produced.

## Part III – Lab Quizz

What is the complexity (in big *O*) of your binary search method for 1) best; 2) average; and 3) worst case?

**Have fun**