Description: In this lab, you will be coding the methods for binary search trees. The first method you will implement is an iterative size computation on a tree. Recall that to compute the size of a tree you will need to use a stack structure and iterate through the leaves of a tree until the stack is empty. See chapter 8.6 (page 554) for the exact details. For this lab, you should use the built-in Java stacks found in java.util.Stack. A reference picture from the book can be seen in Figure 1.

![Figure 1: Pseudocode for an iterative size calculation on a binary tree.](image)

The second operation you will be coding is the add operation. The add is a recursive operation that can be found on page 562 in the book. Although coding this method can be extracted from the book verbatim, it is more important that you understand what is going on with the algorithm. Another reference picture from the book can be seen in Figure 2.

![Figure 2: Pseudocode for a recursive add operation on a binary tree.](image)

Use the template code provided online as a framework for this lab. You can use the included Lab7.java as test driver code to see if your program is working correctly. You will be editing the
ch08/trees/BinarySearchTree.java file, specifically implementing the following methods,

```java
public int size(){
    //implement the iterative size solution here
}

public void add(T element) {
    root = recAdd(element, root);
}
public BSTNode<T> recAdd(T element, BSTNode<T> tree) {
    //implement the recursive add solution here
}
```

Rubric:
(5 points) Compiles without errors.
(20 points) Iterative size correctly implemented with built-in stacks.
(15 points) Recursive add operation correctly implemented.

**Deliverables:** Submit on blackboard.