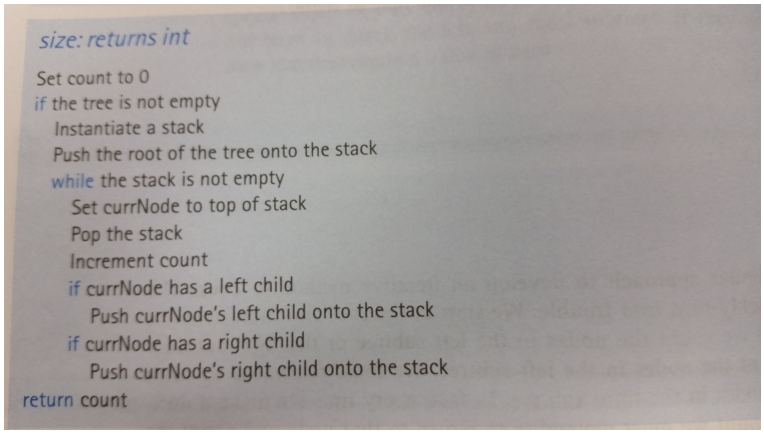
Lab 7
CSC 1052 - Algorithms and Data Structures II
Grading: 40 points
Due Date: April 16th, 2015

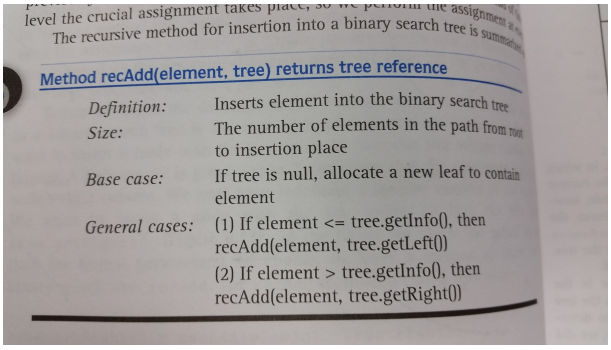
Description: In this lab, you will be coding the methods for binary search trees. The first method you will implement is an iterative size computation on a tree. Recall that to compute the size of a tree you will need to use a stack structure and iterate through the leaves of a tree until the stack is 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.



```
size: returns int
Set count to 0
if the tree is not empty
  Instantiate a stack
  Push the root of the tree onto the stack
  while the stack is not empty
    Set currNode to top of stack
    Pop the stack
    Increment count
    if currNode has a left child
      Push currNode's left child onto the stack
    if currNode has a right child
      Push currNode's right child onto the stack
return count
```

Figure 1: Pseudocode for an iterative size calculation on a binary tree.

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.



level the crucial assignment takes place, so we perform the assignment at
The recursive method for insertion into a binary search tree is summarized

Method recAdd(element, tree) returns tree reference

Definition:	Inserts element into the binary search tree
Size:	The number of elements in the path from root to insertion place
Base case:	If tree is null, allocate a new leaf to contain element
General cases:	(1) If element <= tree.getInfo(), then recAdd(element, tree.getLeft())
	(2) If element > tree.getInfo(), then recAdd(element, tree.getRight())

Figure 2: Pseudocode for a recursive add operation on a binary tree.

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,

```
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.