

Binary Tree Exercise

Instructions

The following Java class definition for a binary search tree will be used to answer the questions below. Unlike project #3, the following tree is not polymorphic and we use null to represent an empty tree. For example, an empty BinarySearchTree has a null root, and a leaf node has null left and right fields.

```
public class BinarySearchTree <K extends Comparable<K>, V> {
    private class Node {
        private K key;
        private V data;
        private Node left, right;
        public Node(K key, V data) {
            this.key = key;
            this.data = data;
        }
    }
    private Node root;
}
```

Exercises

NOTE: You may not add any instance or static variables to the BinarySearchTree class. Adding auxiliary methods is fine.

1. Define a method named **height()** that returns the height of the tree.
2. Define a **recursive** method **size()** that returns the number of entries in the tree.
3. Define a recursive method named **numberOfLeavesNodes ()** that returns the number of leaf nodes in the tree.
4. Define a recursive method **getIncreasingOrderList()** that returns an ArrayList with the **data** elements of the tree inserted into the list based on increasing key order.
5. Define a **recursive** method named **preOrderTraversal** which returns a string representing a pre-order traversal of the tree.
6. Define a recursive method **isFull()** that returns true if every interior node in the tree has both a left and a right subtree.