

## CMSC132 QUIZ #4 Key (DURATION: 25 MINUTES)

FIRSTNAME, LASTNAME (PRINT IN UPPERCASE):

STUDENT ID (e.g. 123456789):

**There are two problems in this quiz; the second problem is on the reverse side.** Implement the methods for this problem based on the following Java class definitions. You may not add any instance variables nor static variables. **If you use any iteration statement (e.g., while loop, do while, for loop) you will get 0 credit for a problem.**

```
public class BinarySearchTreeSoln<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;
}
```

### **Total Pts (60 pts)**

1. Implement the **RECURSIVE** method **getLeavesData** that adds to the ArrayList parameter the **data** component of leaf nodes of the tree. For this problem you can add data to the ArrayList in any order, a tree with only one node has one leaf, the list parameter will never be null, and you may only add one auxiliary method.

One Possible Answer:

```
public void getLeavesData(ArrayList<V> list) {
    getLeavesData(list, root);
}

private void getLeavesData(ArrayList<V> list, Node rootAux) {
    if (rootAux != null) {
        getLeavesData(list, rootAux.left);

        if (rootAux.left == null && rootAux.right == null)
            list.add(rootAux.data);

        getLeavesData(list, rootAux.right);
    }
}
```

2. Implement the **RECURSIVE** method **getKeyNodesAtLevel** that returns an ArrayList with the **key** component of nodes found at the level specified by the **targetLevel** parameter. For this problem you can assume the root is at level 1 and the targetLevel parameter will be greater than or equal to 1. You may only add one auxiliary method.

One Possible Answer:

```
public ArrayList<K> getKeyNodesAtLevel(int targetLevel) {
    ArrayList<K> list = new ArrayList<K>();

    getKeyNodesAtLevel(list, root, 1, targetLevel);

    return list;
}

private void getKeyNodesAtLevel(ArrayList<K> list, Node rootAux,
                                int currLevel, int targetLevel) {
    if (rootAux != null) {
        if (currLevel == targetLevel) {
            list.add(rootAux.key);
        } else {
            getKeyNodesAtLevel(list, rootAux.left, currLevel + 1, targetLevel);
            getKeyNodesAtLevel(list, rootAux.right, currLevel + 1, targetLevel);
        }
    }
}
```