CS 61B Search — Binary Trees and Backtracking — Fall 2014

1 Binary Tree Traversals

```
Preorder: 10, 2, 5, 7, 8, 3, 0, 2
Inorder: 5, 2, 7, 8, 10, 3, 0, 2
Postorder: 5, 8, 7, 2, 2, 0, 3, 10
```

Bonus answers: The height in this case is N. If the preorder and the inorder traversal are the same, this means that all nodes have only right children. The second question is a trick question: Inorder traversal is not well defined for non-binary trees.

2 Is This a BST?

(a) Is the above method correct for all inputs? Why or why not? No, the above code only enforces the BST invariant on at most three nodes at a time, and does not enforce the invariant for ALL keys in the left or right subtrees.

(b) If you answered no to the previous part, what changes would you need to make to isBST() so that it works correctly?

```
public static boolean isBST(TreeNode n) {
    return isBSTHelper(n, Integer.MIN_VALUE, Integer.MAX_VALUE);
}

public static boolean isBSTHelper(TreeNode n, int min, int max) {
    if (n == null) {
        return true;
    }

    if (n.val < min || n.val > max) {
        return false;
    }

    return isBSTHelper(n.left, min, n.val) && isBSTHelper(n.right,
       n.val, max);
}
```

3 Sum Paths

```
void printSumPaths(Node t, int n) {
    if (t != null) {
        sumPathsHelper(t, 0, "", n);
    }
}

void sumPathsHelper(Node curNode, int curSum, String curPath, int n) {
    curSum += curNode.value;
    curPath += curNode.value + " ";
```

```
    if (curNode.left == null && curNode.right == null) {
        if (curSum == n) {
            System.out.println(curPath);
        }
        return;
    }

    if (curNode.left != null) {
        sumPathsHelper(curNode.left, curSum, curPath, n);
    }

    if (curNode.right != null) {
        sumPathsHelper(curNode.right, curSum, curPath, n);
    }
}
```

Bonus question solutions: In the worst **case**, the tree height is Q. At level h, the code performs a concatenation of strings of length k1 * h + k2, e.g.

```
"5"
"5 " + "33"
"5 33 " + "91"
"5 33 91 " + "10"
```

Since String concatenation takes linear time, **this** results in a runtime of $1+2+3+...+Q = \Theta(Q^2)$.