```java
/* Given a binary tree and a sum, find all root-to-leaf paths where each path's sum equals the given sum.

Note: A leaf is a node with no children.

Example:

Given the below binary tree and sum = 22,

      5
     / \
    4   8
   /   / \
  11  13  4
 /  \    / \
7    2  5   1
Return:

[
   [5,4,11,2],
   [5,8,4,5]
] */


import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}

class Solution {

    private List<List<Integer>> result = new ArrayList<>();

    public List<List<Integer>> pathSum(TreeNode root, int sum) {

        if (root == null) {
            return result;
        }

        List<Integer> currentResult = new ArrayList<>();
        pathSumHelper(root, sum, currentResult);

        return result;
    }

    private void pathSumHelper(TreeNode root, int sum, List<Integer> currentResult) {

        currentResult.add(root.val);

        if (root.left == null && root.right == null) {
            if (root.val == sum) {
                List<Integer> currentResultCopy = new ArrayList<>(currentResult);
                result.add(currentResultCopy);
            }
            return;
        }

        if (root.left != null) {
            pathSumHelper(root.left, sum - root.val, currentResult);
            currentResult.remove(currentResult.size() - 1);
        }

        if (root.right != null) {
            pathSumHelper(root.right, sum - root.val, currentResult);
```

```
73                currentResult.remove(currentResult.size() - 1);
74            }
75        }
76    }
```