AROR UNIVERSITY OF ART, ARCHITECTURE, DESIGN & HERITAGE SUKKUR

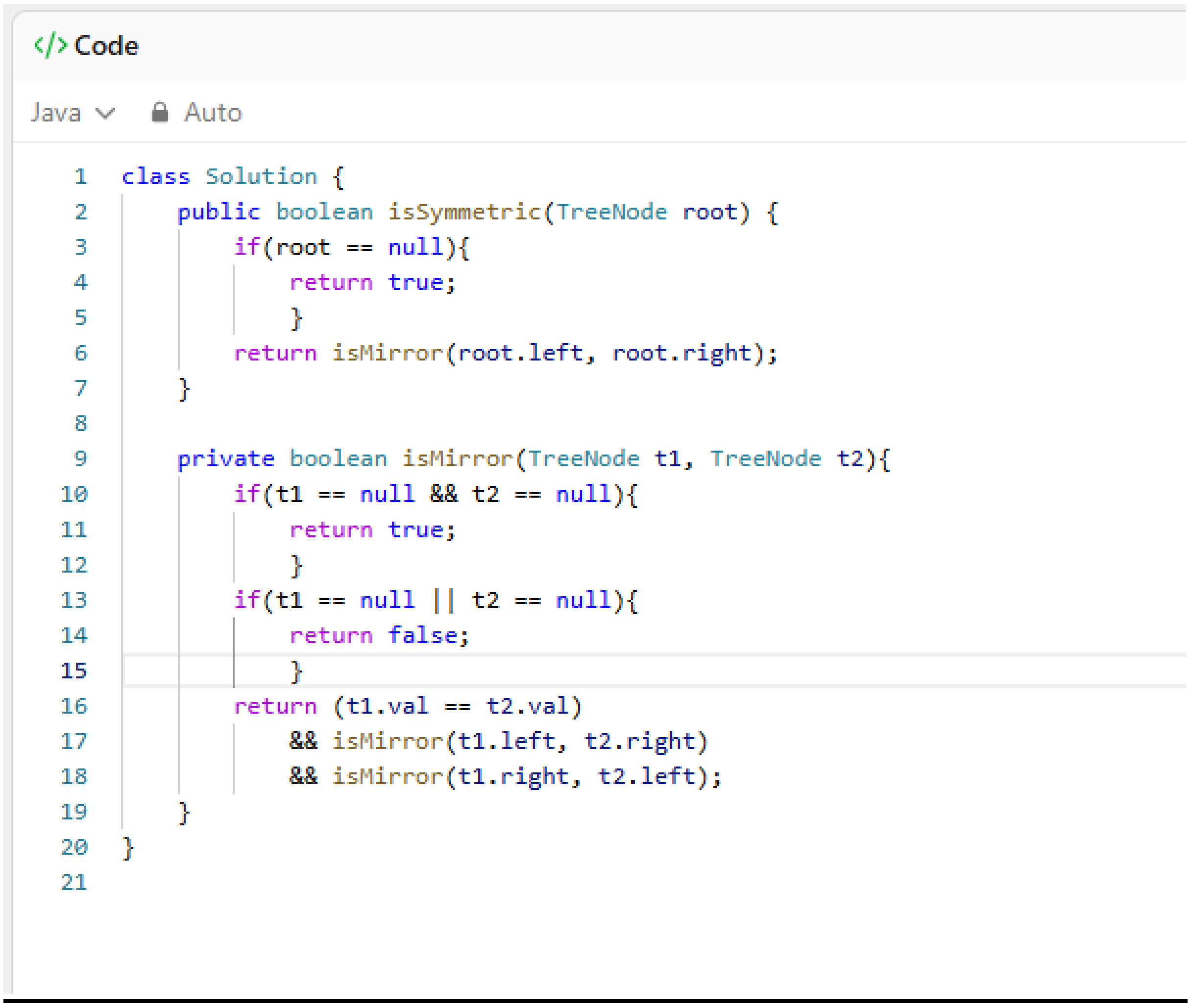
COURSE: Data Structure
BS-Artificial Intelligence (Section B)
LAB # 10

Submitted by: Ibrar Ali
Roll No: 0099

Submitted to: Sir, Abdul Ghafoor

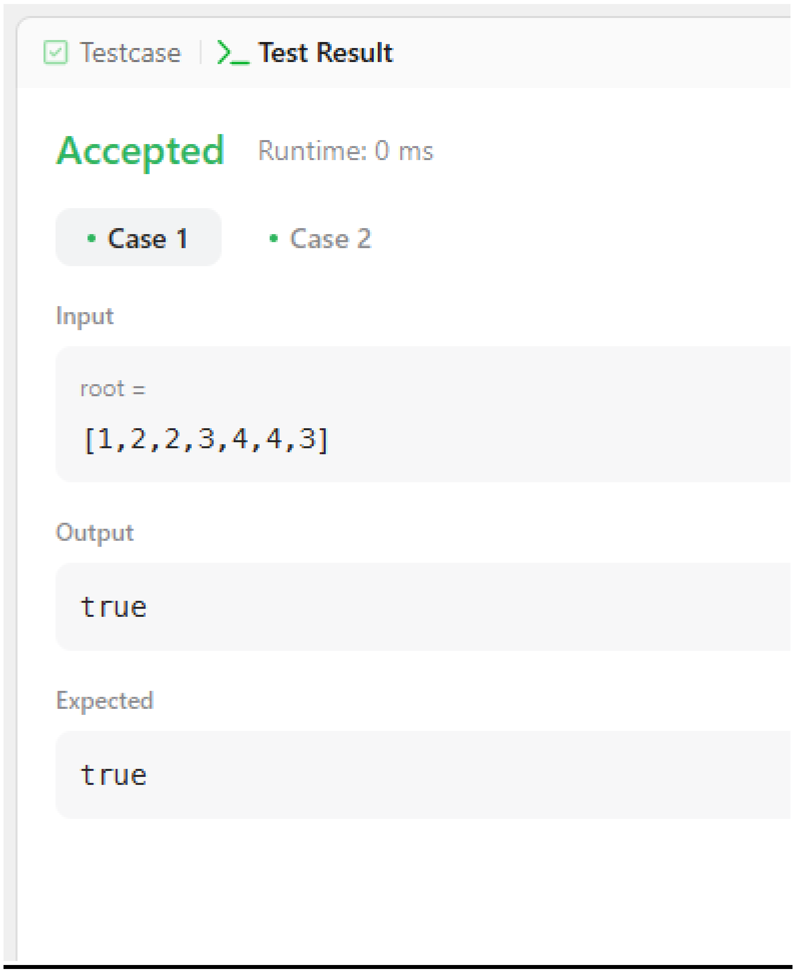
TASK 01: **Symmetric Tree**

Coding:



```java
class Solution {
    public boolean isSymmetric(TreeNode root) {
        if(root == null){
            return true;
            }
        return isMirror(root.left, root.right);
    }

    private boolean isMirror(TreeNode t1, TreeNode t2){
        if(t1 == null && t2 == null){
            return true;
            }
        if(t1 == null || t2 == null){
            return false;
            }
        return (t1.val == t2.val)
            && isMirror(t1.left, t2.right)
            && isMirror(t1.right, t2.left);
    }
}
```

OUTPUT:



Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

```
root =
[1,2,2,3,4,4,3]
```

Output

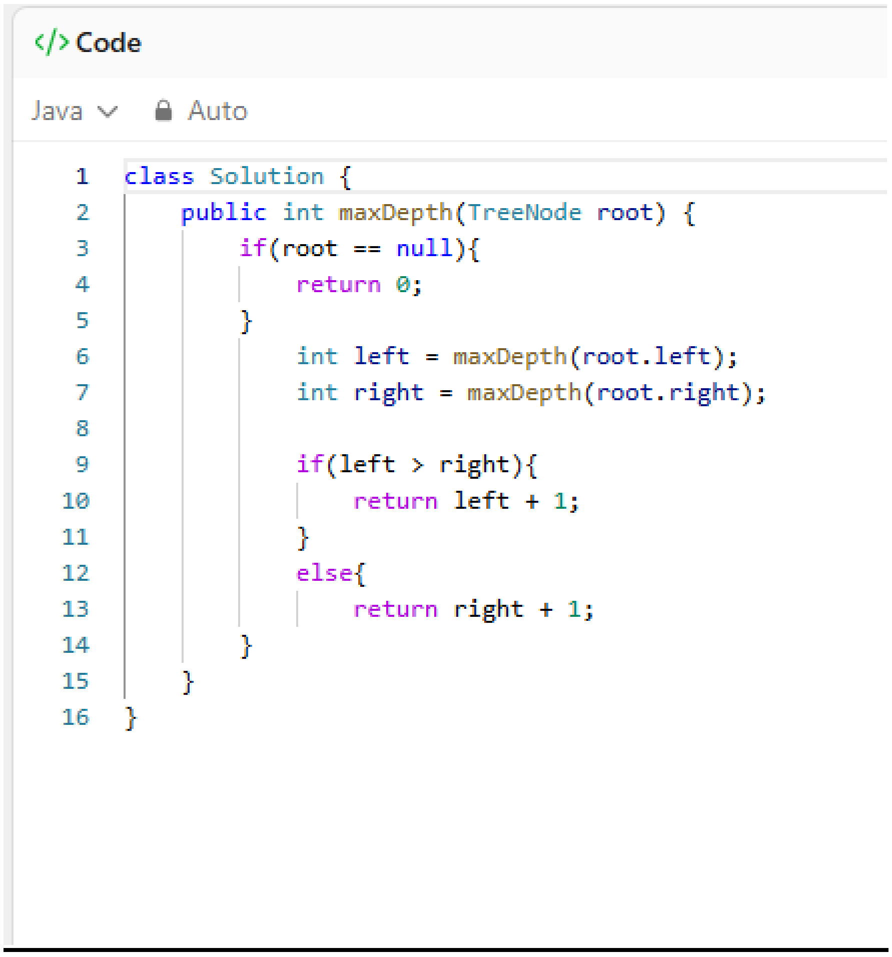
```
true
```

Expected

```
true
```

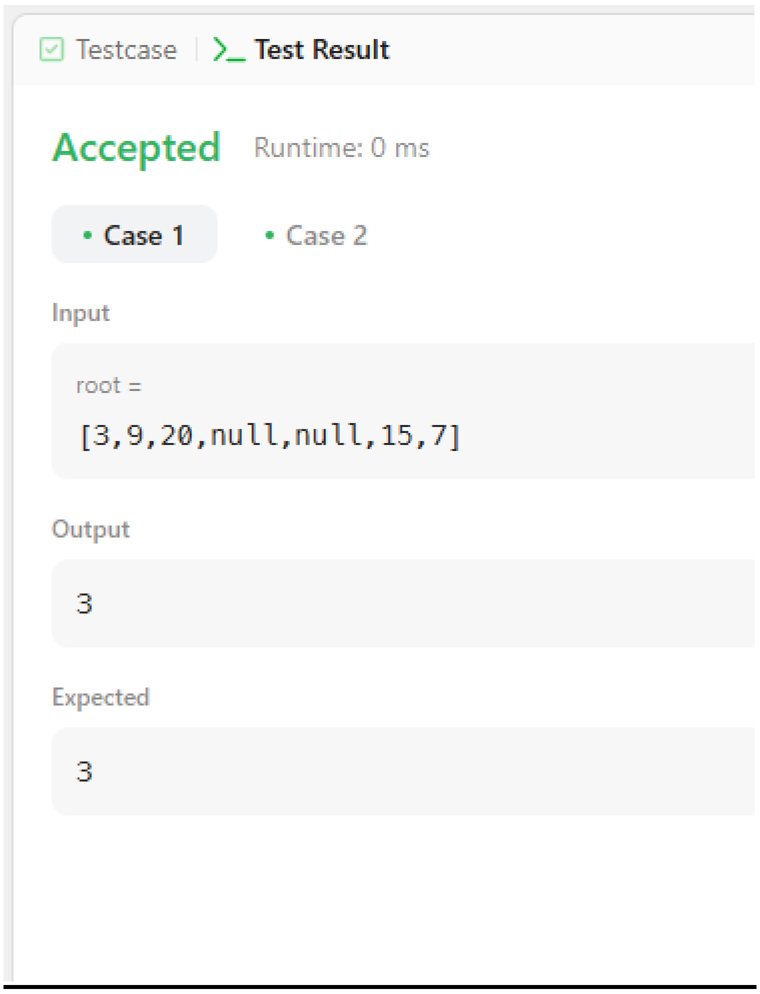
TASK 02: Maximum Depth of Binary Tree

Coding:



```java
class Solution {
    public int maxDepth(TreeNode root) {
        if(root == null){
            return 0;
        }
            int left = maxDepth(root.left);
            int right = maxDepth(root.right);

            if(left > right){
                return left + 1;
            }
            else{
                return right + 1;
        }
    }
}
```

OUTPUT:


Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

```
root =
[3,9,20,null,null,15,7]
```

Output

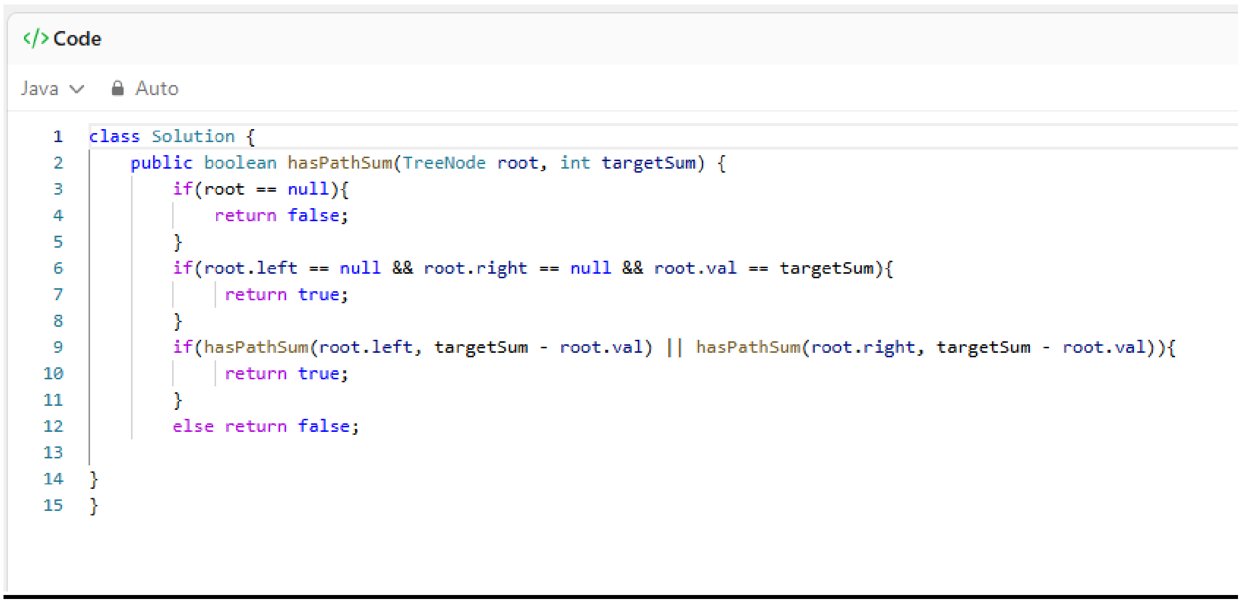
```
3
```

Expected

```
3
```

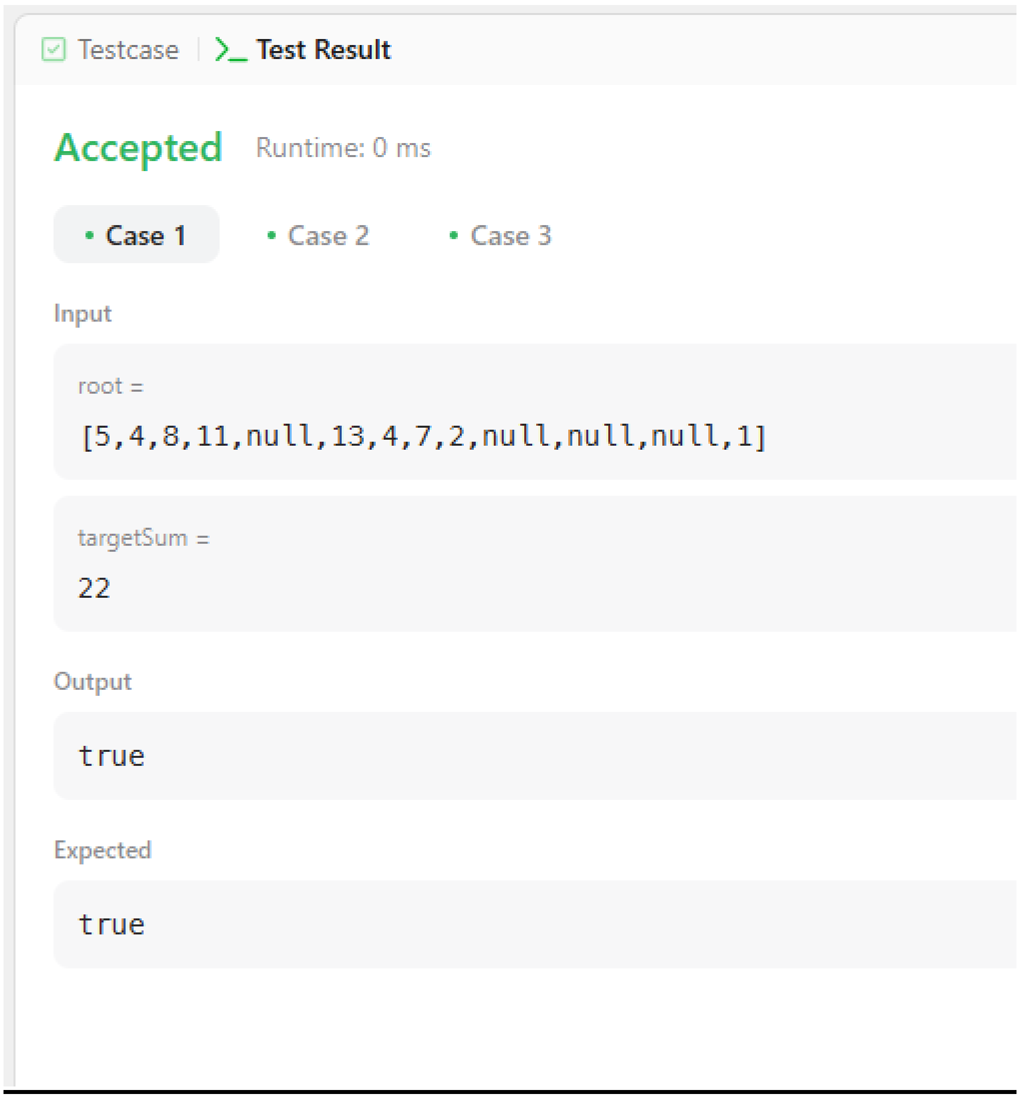
TASK 03: **Path Sum**

Coding:



```java
class Solution {
    public boolean hasPathSum(TreeNode root, int targetSum) {
        if(root == null){
            return false;
        }
        if(root.left == null && root.right == null && root.val == targetSum){
            return true;
        }
        if(hasPathSum(root.left, targetSum - root.val) || hasPathSum(root.right, targetSum - root.val)){
            return true;
        }
        else return false;

}
}
```

OUTPUT:



Accepted Runtime: 0 ms

Input

root =

```
[5,4,8,11,null,13,4,7,2,null,null,null,1]
```

targetSum =

```
22
```

Output

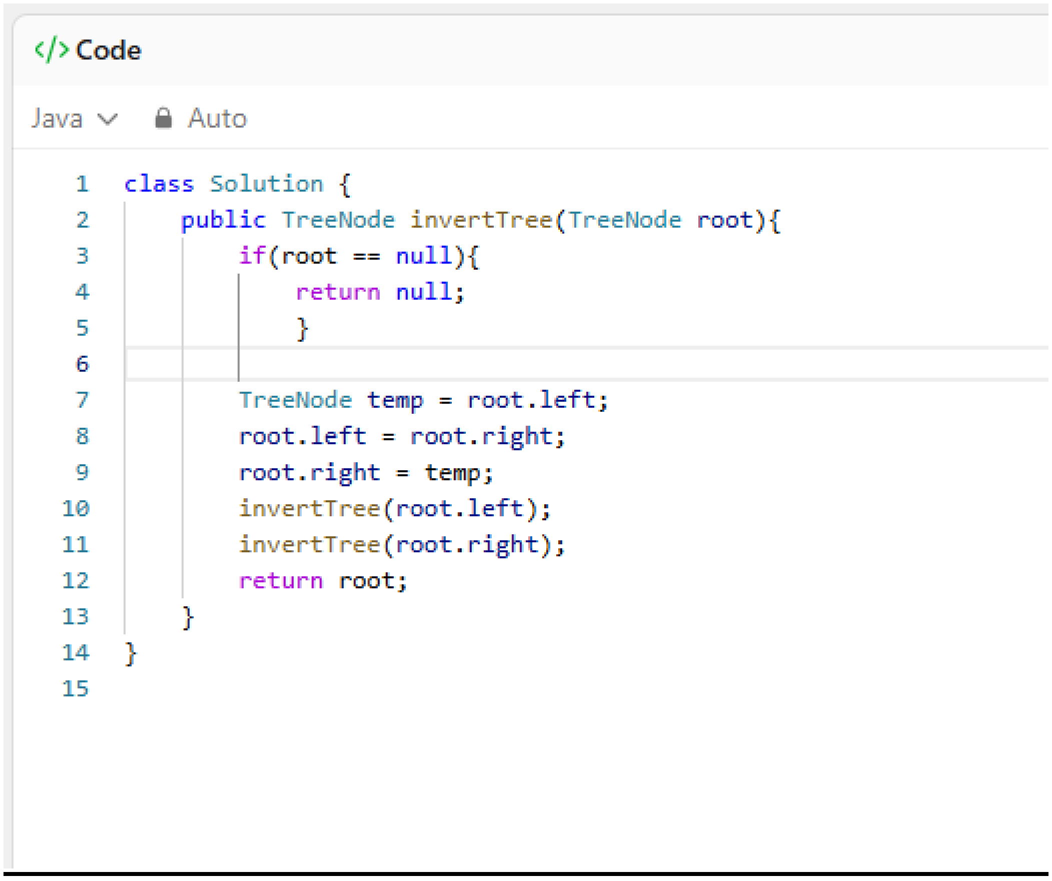
```
true
```

Expected

```
true
```

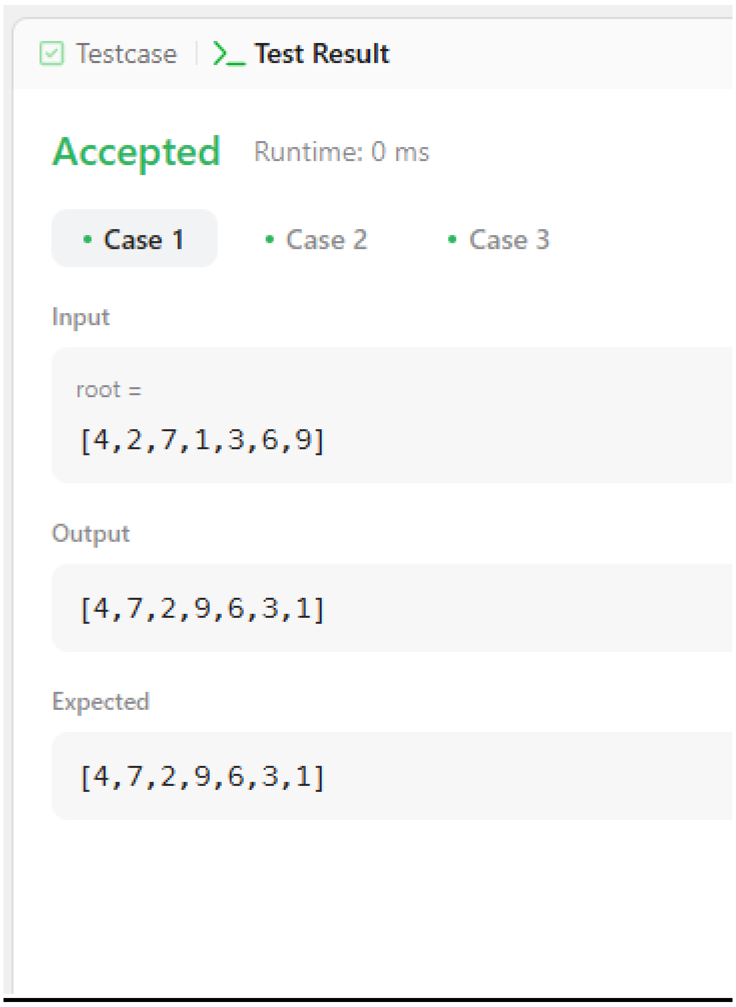
TASK 04: Invert Binary Tree

Coding:



```java
class Solution {
    public TreeNode invertTree(TreeNode root){
        if(root == null){
            return null;
            }

        TreeNode temp = root.left;
        root.left = root.right;
        root.right = temp;
        invertTree(root.left);
        invertTree(root.right);
        return root;
    }
}
```

OUTPUT:



Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

root =

```
[4,2,7,1,3,6,9]
```

Output

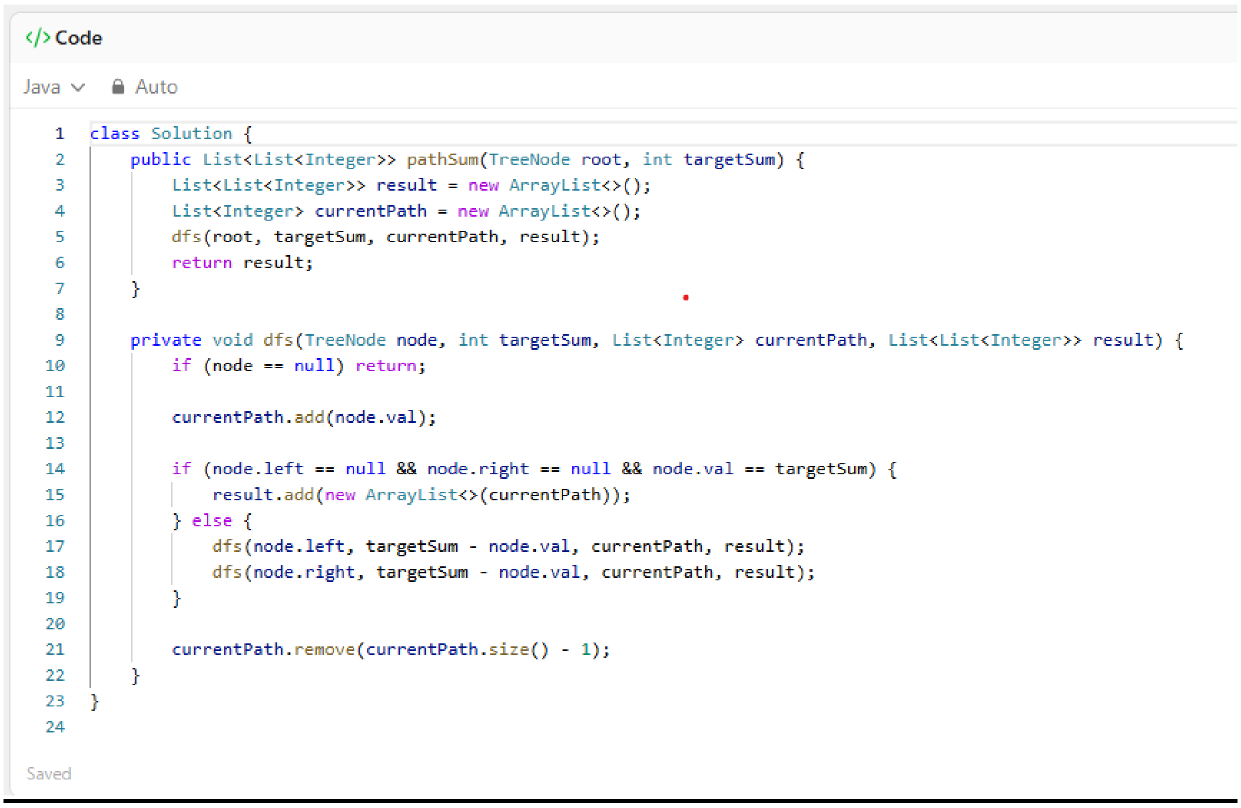
```
[4,7,2,9,6,3,1]
```

Expected

```
[4,7,2,9,6,3,1]
```

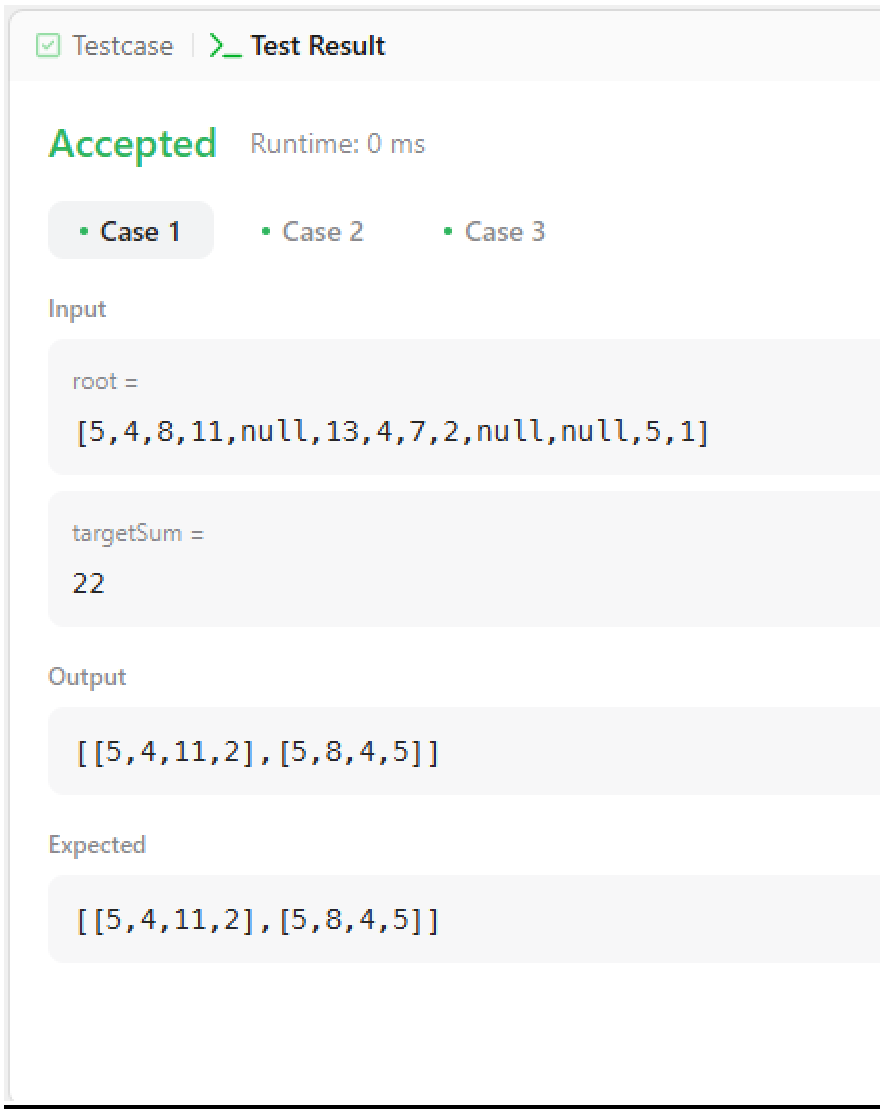
TASK 05: Path Sum II

Coding:



```java
class Solution {
    public List<List<Integer>> pathSum(TreeNode root, int targetSum) {
        List<List<Integer>> result = new ArrayList<>();
        List<Integer> currentPath = new ArrayList<>();
        dfs(root, targetSum, currentPath, result);
        return result;
    }

    private void dfs(TreeNode node, int targetSum, List<Integer> currentPath, List<List<Integer>> result) {
        if (node == null) return;

        currentPath.add(node.val);

        if (node.left == null && node.right == null && node.val == targetSum) {
            result.add(new ArrayList<>(currentPath));
        } else {
            dfs(node.left, targetSum - node.val, currentPath, result);
            dfs(node.right, targetSum - node.val, currentPath, result);
        }

        currentPath.remove(currentPath.size() - 1);
    }
}
```

OUTPUT:



Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

root =

```
[5,4,8,11,null,13,4,7,2,null,null,5,1]
```

targetSum =

```
22
```

Output

```
[[5,4,11,2],[5,8,4,5]]
```

Expected

```
[[5,4,11,2],[5,8,4,5]]
```

THE END