DSA Lab 10

M.Shoaib Lashari
ARI-F23-0081 AI section B

TASK 01

Code

Java Auto

```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);
    }
}
```

TASK 02

Code

Java Auto

```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;
        }
    }
}
```

TASK 03

Code

Java Auto

```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;

}
}
```

TASK 04

Code

Java Auto

```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;
    }
}

```

TASK 05

Code

Java Auto

```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);
    }
}

```

Saved