NAME: SANIA MEMON

ROLL NO: 0091

AI SECTION B

LAB10: DSA

TASK01:

Code

Auto

```java
class Solution {
    public boolean isSymmetric(TreeNode root) {
        //base case
        if(root==null){
            return true;
        }
        return checksymmetric(root.left,root.right);
}
public boolean checksymmetric(TreeNode left,TreeNode right){
    if(left==null && right==null){
        return true;
    }
    if(left==null || right==null || left.val != right.val){
        return  false;
    }
    return checksymmetric(left.left,right.right) && checksymmetric(left.right,
right.left);
}
}
```

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

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

Output

true

Expected

true

TASK02:

```
 *      int val;
 *      TreeNode left;
 *      TreeNode right;
 *      TreeNode() {}
 *      TreeNode(int val) { this.val = val; }
 *      TreeNode(int val, TreeNode left, TreeNode right) {
 *          this.val = val;
 *          this.left = left;
 *          this.right = right;
 *      }
 * }
 */
class Solution {
    public int maxDepth(TreeNode root) {
        if(root==null){
            return 0;
        }
        int l = maxDepth(root.left);
        int r = maxDepth(root.right);
        return 1+Math.max(l,r);
    }
}
```

TASK03:

```
 * }
 */
class Solution {
    public boolean hasPathSum(TreeNode root, int targetSum) {
        //base case
        if(root==null){
            return false;
        }
        targetSum -= root.val;
        if(root.left==null && root.right==null){
            return targetSum==0;
        }


        return hasPathSum(root.right,targetSum) ||  hasPathSum(root.left,
targetSum) ;
    }
}
```

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

root =

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

targetSum =

22

Output

true

TASK04:

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

        // Swap the left and right children
        TreeNode temp = root.left;
        root.left = root.right;
        root.right = temp;

        // Recursively invert the left and right subtrees
        invertTree(root.left);
        invertTree(root.right);

        return root;
    }
```

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]

TASK05:

```
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;
        }

        // Add the current node's value to the path
        currentPath.add(node.val);

        // Check if we are at a leaf and the target sum is met
        if (node.left == null && node.right == null && node.val == targetSum) {
            result.add(new ArrayList<>(currentPath));
        result.add(new ArrayList<>(currentPath));
    } else {
        // Continue to explore the left and right subtrees
        dfs(node.left, targetSum - node.val, currentPath, result);
        dfs(node.right, targetSum - node.val, currentPath, result);
    }

    // Backtrack: Remove the last node from the path
    currentPath.remove(currentPath.size() - 1);
```

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