TREES IN-CLASS PROBLEMS :

(Mentor : Gladden Rumao)

1. **Count Leaves in Binary Tree**
   https://practice.geeksforgeeks.org/problems/count-leaves-in-binary-tree/1

```
class Tree
{
    int countLeaves(Node node)
    {
        if(node==null)
            return 0;
        if(node.left==null && node.right==null)
            return 1;
        return
countLeaves(node.left)+countLeaves(node.right);
    }
}
```

2. **Binary Tree Preorder Traversal**
   https://leetcode.com/problems/binary-tree-preorder-traversal/

```
class Solution {
    List<Integer> L = new ArrayList<>();
    public List<Integer>
preorderTraversal(TreeNode root) {
        if(root==null)
            return L;
        L.add(root.val);
        preorderTraversal(root.left);
        preorderTraversal(root.right);
```

```
        return L;
    }
}
```

3. **Binary Tree Inorder Traversal**
   https://leetcode.com/problems/binary-tree-preorder-traversal/

```
class Solution {
    List<Integer> L = new ArrayList<>();
    public List<Integer> inorderTraversal(TreeNode
root) {
        if(root==null)
            return L;
        inorderTraversal(root.left);
        L.add(root.val);
        inorderTraversal(root.right);
        return L;
    }
}
```

4. **Binary Tree Postorder Traversal**
   https://leetcode.com/problems/binary-tree-postorder-traversal/

```
class Solution {
    List<Integer> L = new ArrayList<>();
    public List<Integer>
postorderTraversal(TreeNode root) {
        if(root==null){
            return L;
        }

        postorderTraversal(root.left);
```

```
        postorderTraversal(root.right);
        L.add(root.val);

        return L;

    }
}
```

5. **Height of Binary Tree**
   https://leetcode.com/problems/maximum-depth-of-binary-tree/

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

6. **Level Order Traversal**
   https://practice.geeksforgeeks.org/problems/level-order-traversal/1

```
class Solution
{
    //Function to return the level order traversal
of a tree.
    static ArrayList <Integer> levelOrder(Node
node)
    {
        ArrayList<Integer> ans = new
ArrayList<>();
```

```
        if(node==null)
            return ans;

        Queue<Node> q = new LinkedList<>();
        q.add(node);

        while(!q.isEmpty()){
            Node n = q.remove();
            ans.add(n.data);

            if(n.left!=null)
                q.add(n.left);
            if(n.right!=null)
                q.add(n.right);
        }
        return ans;

    }
}
```

7. **Diameter of Tree**
   https://leetcode.com/problems/diameter-of-binary-tree/

```
class Solution {
    int max = 0;
    public int diameterOfBinaryTree(TreeNode root)
{
        height(root);
        return max;
    }
    public int height(TreeNode root){
```

```
        if(root==null)
            return 0;

        int left = height(root.left);
        int right = height(root.right);

        max = Math.max(max , left+right);

        return 1+Math.max(left,right);
    }
}
```

8. **Symmetric Tree**
   https://leetcode.com/problems/symmetric-tree/

```
class Solution {
    public boolean check(TreeNode left , TreeNode
right){
        if(left==null && right==null)
            return true;
        if(left==null || right==null)
            return false;
        if(left.val!=right.val)
            return false;

        return check(left.left,right.right) &&
check(left.right,right.left);
    }
    public boolean isSymmetric(TreeNode root) {
        return check(root,root);
    }
```

```
}
```

9. **Maximum Path Sum of Binary Tree**
   https://leetcode.com/problems/binary-tree-maximum-path-sum/

```
class Solution {
    int max = Integer.MIN_VALUE;

    public int maxPathSum(TreeNode root) {
        helper(root);
        return max;
    }

    public int helper(TreeNode root) {
        if(root == null)
            return 0;
        int left = Math.max(0, helper(root.left));
        int right = Math.max(0,
helper(root.right));
        max = Math.max(max, root.val + left +
right);
        return root.val + Math.max(left, right);
    }
}
```

10. **Lowest Common Ancestors of Binary Tree**
    https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/

```
public class Solution {
    public TreeNode lowestCommonAncestor(TreeNode
root, TreeNode p, TreeNode q) {
        if(root == null)
```

```
            return null;
        if(root == p || root == q)
            return root;

        TreeNode left =
lowestCommonAncestor(root.left, p, q);
        TreeNode right =
lowestCommonAncestor(root.right, p, q);

        if(left != null && right != null)
            return root;
        if (left != null)
            return left;
        else
            return right;

    }
}
```