Tree

Problem Statement 1
Create a Binary Tree and Print the level order traversal of a binary tree..
Input: take input as level order.
Output: Print the Level order traversal.

PROGRAM:

```
import java.util.*;
class Node {
    int data;
    Node left, right;
    Node(int val) {
        this.data = val;
        this.left = this.right = null;
    }
}
 class LevelOrderTree {
    public static Node buildTreeFromArray(int[] arr) {
        if (arr.length == 0 || arr[0] == -1) return null;
        Node root = new Node(arr[0]);
        Queue<Node> q = new LinkedList<>();
        q.offer(root);
        int i = 1;
        while(!q.isEmpty() && i < arr.length) {
            Node current = q.poll();
            if (i < arr.length && arr[i] != -1) {
                current.left = new Node(arr[i]);
                q.offer(current.left);
            }
            i++;
            if (i < arr.length && arr[i] != -1) {
                current.right = new Node(arr[i]);
                q.offer(current.right);
            }
            i++;
        }

        return root;
    }
    public static void inorder(Node root) {
        if (root == null) return;
        inorder(root.left);
        System.out.print(root.data + " ");
        inorder(root.right);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of elements: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter " + n + " elements (-1 for null):");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        Node root = buildTreeFromArray(arr);
        System.out.println("Inorder traversal of constructed tree:");
        inorder(root);
    }
}
```

Problem Statement 2
Count the number of leaf nodes in a binary tree.

Programs:

```
import java.util.*;
class Node {
    int data;
    Node left, right;
    Node(int d) { data = d; }
}
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] a = new int[n];
        for (int i = 0; i < n; i++) a[i] = sc.nextInt();
        Queue<Node> q = new LinkedList<>();
        Node root = new Node(a[0]);
        q.add(root);
        int i = 1;
        while (!q.isEmpty() && i < n) {
            Node cur = q.poll();
            if (a[i] != -1) {
                cur.left = new Node(a[i]);
                q.add(cur.left);
            }
            i++;
            if (i < n && a[i] != -1) {
                cur.right = new Node(a[i]);
                q.add(cur.right);
            }
            i++;
        }
        System.out.println(countLeaves(root));
    }
    static 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);
    }
}
```

Problem Statement 3
Find the height (or depth) of a binary tree.

Program:

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;
class Node {
    int data;
    Node left, right;
    Node(int data) {
        this.data = data;
        this.left = this.right = null;
    }
}
public class Main {
    public static Node buildTree(int[] values) {
        if (values.length == 0 || values[0] == -1) return null;
        Node root = new Node(values[0]);
        Queue<Node> queue = new LinkedList<>();
        queue.offer(root);
        int i = 1;
        while (!queue.isEmpty() && i < values.length) {
            Node current = queue.poll();
            if (values[i] != -1) {
                current.left = new Node(values[i]);
                queue.offer(current.left);
            }
            i++;
            if (i < values.length && values[i] != -1) {
                current.right = new Node(values[i]);
                queue.offer(current.right);
            }
            i++;
        }
        return root;
    }
    public static int findHeight(Node root) {
        if (root == null) return -1;
        Queue<Node> queue = new LinkedList<>();
        queue.offer(root);
        int height = -1;
        while (!queue.isEmpty()) {
            int levelSize = queue.size();
            height++;
            for (int i = 0; i < levelSize; i++) {
                Node current = queue.poll();
                if (current.left != null) queue.offer(current.left);
                if (current.right != null) queue.offer(current.right);
            }
        }
        return height;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] values = new int[n];
        for (int i = 0; i < n; i++) values[i] = sc.nextInt();
        Node root = buildTree(values);
        System.out.println(findHeight(root));
    }
}
```