**Assignment17**

1. **Level Order Traversal**

import java.util.\*;

class Node {

int data;

Node left, right;

Node(int data) {

this.data = data;

}

}

public class BinaryTreeLevelOrder {

public static Node buildTreeLevelOrder(Integer[] arr) {

if (arr.length == 0 || arr[0] == null)

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 curr = q.poll();

if (i < arr.length && arr[i] != null) {

curr.left = new Node(arr[i]);

q.offer(curr.left);

}

i++;

if (i < arr.length && arr[i] != null) {

curr.right = new Node(arr[i]);

q.offer(curr.right);

}

i++;

}

return root;

}

public static void printLevelOrder(Node root) {

if (root == null) return;

Queue<Node> q = new LinkedList<>();

q.offer(root);

while (!q.isEmpty()) {

Node curr = q.poll();

System.out.print(curr.data + " ");

if (curr.left != null) q.offer(curr.left);

if (curr.right != null) q.offer(curr.right);

}

}

public static void main(String[] args) {

Integer[] input = {1, 2, 3, null, 4, 5, null}; // Sample input

Node root = buildTreeLevelOrder(input);

System.out.print("Level Order Traversal: ");

printLevelOrder(root);

}

}

### 2.Count Leaf Nodes

public class CountLeafNodes {

public static int countLeaves(Node root) {

if (root == null)

return 0;

if (root.left == null && root.right == null)

return 1;

return countLeaves(root.left) + countLeaves(root.right);

}

public static void main(String[] args) {

Integer[] input = {1, 2, 3, null, 4, 5, null};

Node root = BinaryTreeLevelOrder.buildTreeLevelOrder(input);

int count = countLeaves(root);

System.out.println("Number of Leaf Nodes: " + count);

}

}

**3.Height of Binary Tree**

public class TreeHeight {

public static int findHeight(Node root) {

if (root == null)

return 0;

return 1 + Math.max(findHeight(root.left), findHeight(root.right));

}

public static void main(String[] args) {

Integer[] input = {1, 2, 3, null, 4, 5, null};

Node root = BinaryTreeLevelOrder.buildTreeLevelOrder(input);

int height = findHeight(root);

System.out.println("Height of Tree: " + height);

}

}