1. Find out if a Tree is Balanced. (Tree is balanced if for every node difference between height of left and right is <=1)

class Node{

int data;

Node left,right;

Node(int data){

this.data=data;

}

}

public class BinaryTree\_BalancedorNot {

Node root;

int height(Node node){

if(node==null)

return 0;

else

return 1 + Math.max(height(node.left),height(node.right));

}

boolean CheckIfBalanced(Node node){

int left\_height, right\_height;

if(node==null)

return true;

left\_height=height(node.left);

right\_height=height(node.right);

if(Math.abs(left\_height-right\_height)<=1 && CheckIfBalanced(node.left) && CheckIfBalanced(node.right))

return true;

else

return false;

}

public static void main(String[] args) {

BinaryTree\_BalancedorNot ob=new BinaryTree\_BalancedorNot();

ob.root=new Node(1);

ob.root.left=new Node(2);

ob.root.right=new Node(3);

ob.root.left.left=new Node(4);

ob.root.right.left=new Node(5);

if(ob.CheckIfBalanced(ob.root))

System.out.println("Tree is balanced");

else

System.out.println("Tree is not balanced");

}

}

------------------------------------------------------------------------------------------------------------------------------------------

2. Find Minimum depth of all the leaves in a tree.

class Node{

int data;

Node left, right;

public Node(int data){

this.data=data;

}

}

public class MinimumDepth\_BinaryTree {

Node root;

int min\_depth(){

return min\_depth(root);

}

int min\_depth(Node root){

if(root==null)

return 0;

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

return 1;

if(root.left==null)

return min\_depth(root.right)+1;

if(root.right==null)

return min\_depth(root.left) +1;

return Math.min(min\_depth(root.left),min\_depth(root.right))+1;

}

public static void main(String[] args) {

MinimumDepth\_BinaryTree ob=new MinimumDepth\_BinaryTree();

ob.root=new Node(1);

ob.root.left=new Node(2);

ob.root.right=new Node(3);

ob.root.left.left=new Node(4);

System.out.println("The minimum depth of binary tree is: "+ob.min\_depth());

}

}

------------------------------------------------------------------------------------------------------------------------

3. Print all the leaves of a tree.

import java.util.ArrayList;

import java.util.List;

class Node{

int data;

Node left, right;

Node(int data){

this.data=data;

}

}

public class PrintLeavesOfBinaryTree {

public List<List<Integer>> printLeaves(Node root){

List<List<Integer>> result= new ArrayList<>();

height(result, root);

return result;

}

public int height(List<List<Integer>> result, Node root){

if(root==null){

return -1;

}

int left\_height=height(result,root.left);

int right\_height=height(result,root.right);

int current\_height=Math.max(left\_height,right\_height)+1;

if(result.size()<=current\_height){

result.add(new ArrayList<>());

}

result.get(current\_height).add(root.data);

return current\_height;

}

public static void main(String[] args) {

PrintLeavesOfBinaryTree ob=new PrintLeavesOfBinaryTree();

Node root=new Node(1);

root.left=new Node(2);

root.right=new Node(3);

root.left.left=new Node(4);

root.right.left=new Node(5);

root.right.left.left=new Node(6);

List<List<Integer>> l=ob.printLeaves(root);

System.out.print("[");

for(List<Integer> leaveLevel:l){

System.out.print("[");

for (int leaf\_l:leaveLevel){

System.out.print(leaf\_l+" ");

}

System.out.print("]");

}

System.out.println("]");

}

}

------------------------------------------------------------------------------------------------------------------------------------------ 4. Print sum of all the left leaves.

public class Sum\_LeftLeaves {

static int sum=0;

static class Node{

int data;

Node left,right;}

static Node an(int data){

Node n=new Node();

n.data=data;

n.left=null;

n.right=null;

return n;

}

static void LeftLeavesSum (Node root){

if(root==null)

return;

if(root.left!=null)

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

sum=sum+root.left.data;

LeftLeavesSum(root.left);

LeftLeavesSum(root.right);

}

public static void main(String[] args) {

Node root=an(1);

root.left=an(2);

root.right=an(3);

root.left.left=an(4);

root.right.left=an(5);

root.right.left.left=an(6);

sum=0;

LeftLeavesSum(root);

System.out.println("The sum of left leaves is: "+sum);

}

}