19-March-2020

Link for practice:

https://thecodingsimplified.com/binary-tree/

Code:

```
package mbatch;
import java.util.*;
class Node
{
    int data;
    Node left;
    Node right;
    Node(int data){
        this.data=data;
        left=null;
        right=null;
    }
}
class BinaryTree{
    Node root;
    BinaryTree(){

        root=null;
    }
    BinaryTree(int data){
        this.root=new Node(data);
    }
    int TreeSum(Node root)//to calculate the sum
of all nodes in a tree
```

```
    {
        if(root==null) return 0;
        return
root.data+TreeSum(root.left)+TreeSum(root.right)
;
    }
    int countNodes(Node root)//to calculate the
number of nodes in a tree
    {
        if(root==null) return 0;
        return
1+countNodes(root.left)+countNodes(root.right);
    }
    int leafNodes(Node root)//to calculate the
number of Leaf Nodes in a tree
    {
        if(root==null) return 0;
        if(root.left==null && root.right==null)
return 1;
        return
leafNodes(root.left)+leafNodes(root.right);
    }
    int sumleafNodes(Node root)//to calculate
the sum of Leaf Nodes in a tree
    {
        if(root==null) return 0;
        if(root.left==null && root.right==null)
return root.data;
        return
sumleafNodes(root.left)+sumleafNodes(root.right)
;
    }
    int height(Node root)
    {
        if(root==null) return -1;
```

```
        return 1+Math.max(height(root.left),
height(root.right));
    }
    void printAtLevel(Node root,int level)
    {
        if(root==null) return;
        if(level==1)
        {
            System.out.print(root.data+" ");
            return;
        }
        printAtLevel(root.left,level-1);
        printAtLevel(root.right,level-1);
    }
    void levrec(Node root)
    {
        if(root==null) return;
        int h=height(root);
        for(int i=1;i<=h+1;i++)
        {
            printAtLevel(root, i);
            System.out.println();
        }
    }
    void levitr(Node root)
    {
        if(root==null) return;
        Queue<Node> q=new
java.util.LinkedList<>();
        q.add(root);
        while(!q.isEmpty())
        {
            Node temp=q.remove();
            System.out.print(temp.data+" ");
            if(temp.left!=null)
```

```
            {
                q.add(temp.left);
            }
            if(temp.right!=null)
            {
                q.add(temp.right);
            }
        }
        System.out.println();
    }
    void levlineitr(Node root)
    {
        if(root==null) return;
        Queue<Node> q=new
java.util.LinkedList<>();
        q.add(root);
        while(true)
        {
            int size=q.size();
            if(size==0) break;
            //while(size>0)
            for(int i=0;i<size;i++)
            {
                Node temp=q.remove();
                System.out.print(temp.data+" ");
                if(temp.left!=null)
                {
                    q.add(temp.left);
                }
                if(temp.right!=null)
                {
                    q.add(temp.right);
                }
                //size--;
            }
```

```
            System.out.println();
        }
    }
    boolean isIdentical(Node root1, Node root2)
    {
        if(root1==null && root2==null) return
true;
        if(root1==null||root2==null) return
false;
        return root1.data==root2.data
                &&
isIdentical(root1.left,root2.left)
                &&
isIdentical(root1.right,root2.right);

    }
}
public class btree {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        BinaryTree bt=new BinaryTree(2);//BT with
root node 2
        bt.root.left=new Node(3);//linking
explicitly
        bt.root.right=new Node(5);
        bt.root.left.right=new Node(9);
        bt.root.right.left=new Node(7);//Required
Tree Created
        System.out.println("Sum of all Nodes:
"+bt.TreeSum(bt.root));
        System.out.println("Total Nodes:
"+bt.countNodes(bt.root));
        System.out.println("Leaf Nodes:
"+bt.leafNodes(bt.root));
```

```
        System.out.println("Height:
"+bt.height(bt.root));
        System.out.print("Nodes at level 1: ");
        bt.printAtLevel(bt.root,1);
        System.out.println();
        System.out.print("Nodes at level 2: ");
        bt.printAtLevel(bt.root,2);
        System.out.println();
        System.out.print("Nodes at level 3: ");
        bt.printAtLevel(bt.root,3);
        System.out.println();
        System.out.print("Nodes at level 4: ");
        bt.printAtLevel(bt.root,4);
        System.out.println();
        System.out.println("Sum of Leaf Nodes:
"+bt.sumleafNodes(bt.root));
        bt.levrec(bt.root);
        bt.levitr(bt.root);
        bt.levlineitr(bt.root);

        BinaryTree bt2=new BinaryTree(2);//BT
with root node 2
        bt2.root.left=new Node(3);//linking
explicitly
        bt2.root.right=new Node(5);
        bt2.root.left.right=new Node(8);
        bt2.root.right.left=new Node(7);

    System.out.println(bt.isIdentical(bt.root,
bt2.root));
    }
}
```