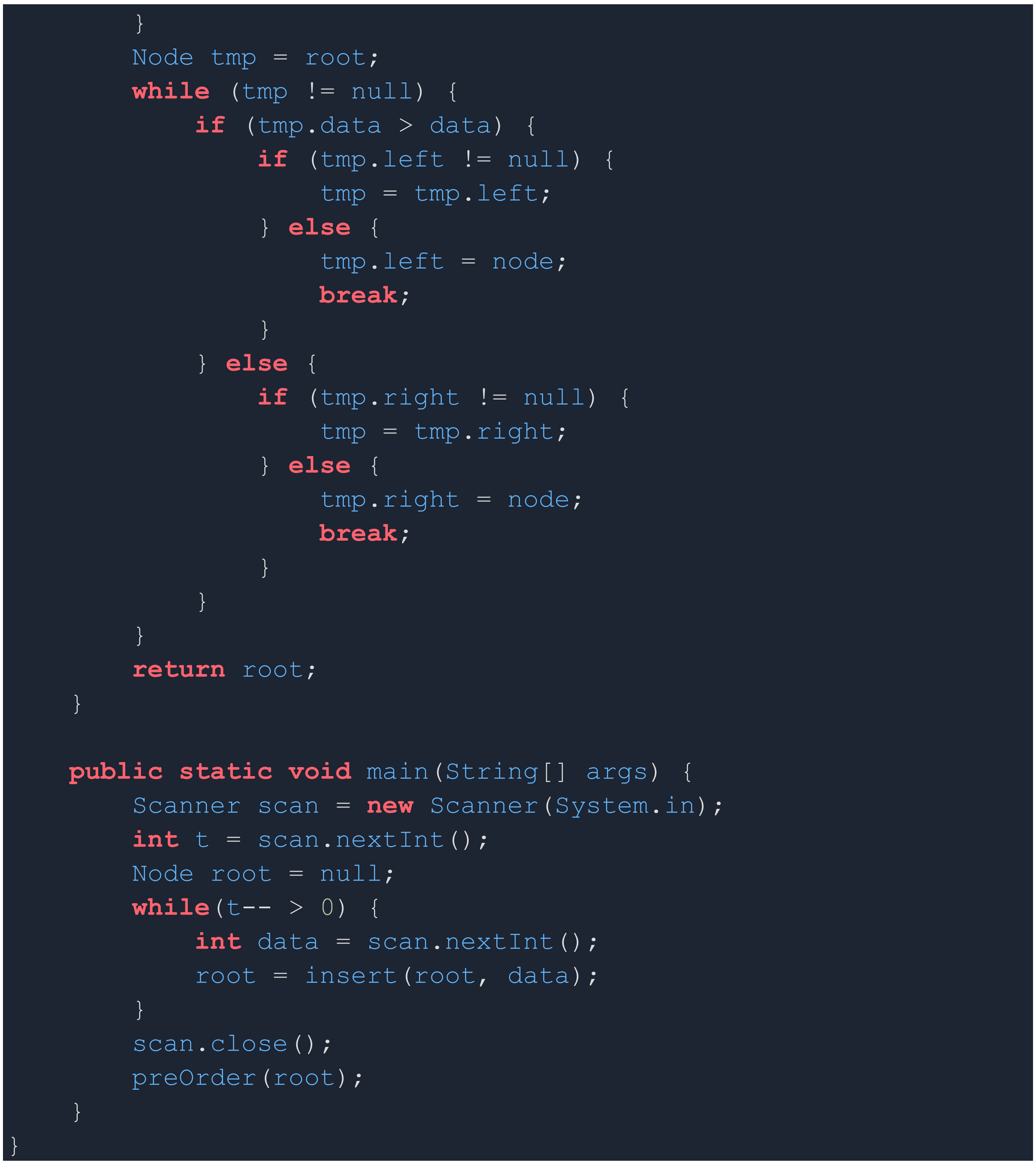
PRACTICAL 10

10.1 Implement Binary Search Tree (BST) Insertion
https://www.hackerrank.com/challenges/binary-search-tree-insertion/problem

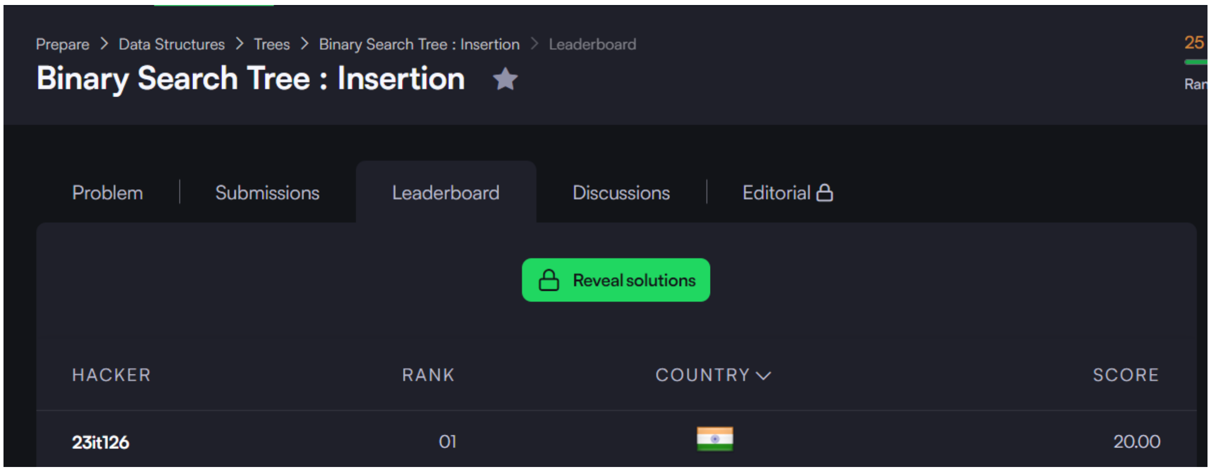
CODE :

```java
import java.util.*;
import java.io.*;

class Node {
    Node left;
    Node right;
    int data;

    Node(int data) {
        this.data = data;
        left = null;
        right = null;
    }
}

class Solution {

    public static void preOrder( Node root ) {

        if( root == null)
            return;

        System.out.print(root.data + " ");
        preOrder(root.left);
        preOrder(root.right);

    }

 /* Node is defined as :
 class Node
    int data;
    Node left;
    Node right;

    */

    public static Node insert(Node root,int data) {
         Node node = new Node(data);
        if (root == null) {
            return node;
```



```
        }
        Node tmp = root;
        while (tmp != null) {
            if (tmp.data > data) {
                if (tmp.left != null) {
                    tmp = tmp.left;
                } else {
                    tmp.left = node;
                    break;
                }
            } else {
                if (tmp.right != null) {
                    tmp = tmp.right;
                } else {
                    tmp.right = node;
                    break;
                }
            }
        }
        return root;
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int t = scan.nextInt();
        Node root = null;
        while(t-- > 0) {
            int data = scan.nextInt();
            root = insert(root, data);
        }
        scan.close();
        preOrder(root);
    }
}
```

OUTPUT:


Prepare > Data Structures > Trees > Binary Search Tree : Insertion > Leaderboard

Binary Search Tree : Insertion

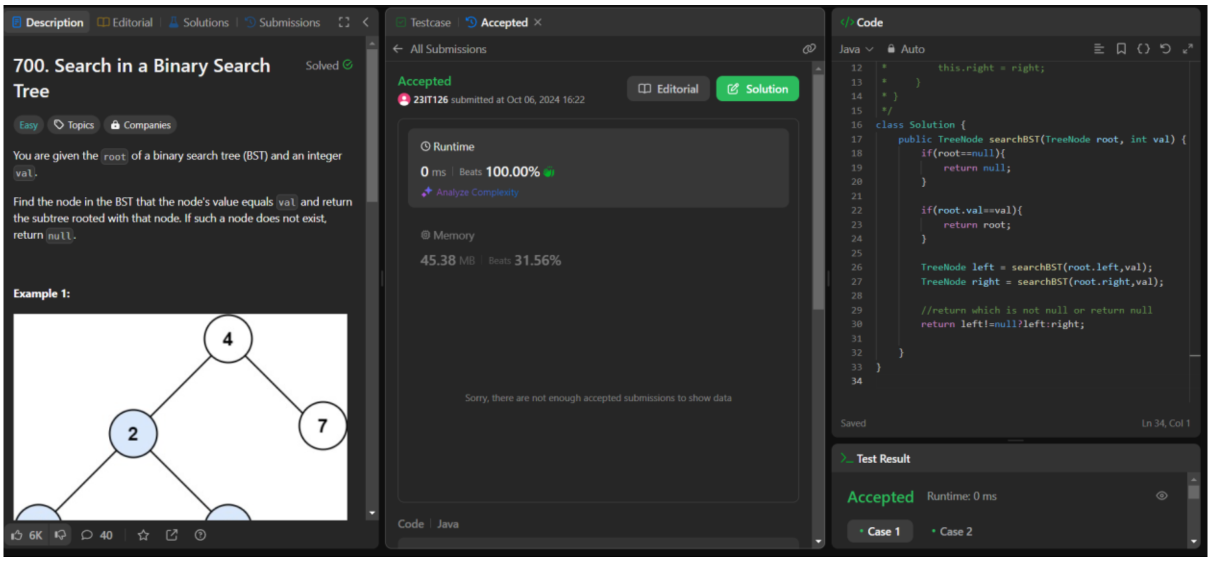
HACKER	RANK	COUNTRY	SCORE
23it126	01		20.00

10.2 Implement searching in Binary Search Tree (BST)
https://leetcode.com/problems/search-in-a-binary-search-tree/

CODE:

```java
class Solution {
    public TreeNode searchBST(TreeNode root, int val) {
        if(root==null){
            return null;
        }

        if(root.val==val){
            return root;
        }

        TreeNode left = searchBST(root.left,val);
        TreeNode right = searchBST(root.right,val);

        //return which is not null or return null
        return left!=null?left:right;

    }
}
```

OUTPUT:



700. Search in a Binary Search Tree

Solved

You are given the `root` of a binary search tree (BST) and an integer `val`.

Find the node in the BST that the node's value equals `val` and return the subtree rooted with that node. If such a node does not exist, return `null`.

Example 1:

Accepted
23IT126 submitted at Oct 06, 2024 16:22

Runtime
0 ms | Beats 100.00%

Memory
45.38 MB | Beats 31.56%

Sorry, there are not enough accepted submissions to show data

```java
 *          this.right = right;
 *      }
 * }
 */
class Solution {
    public TreeNode searchBST(TreeNode root, int val) {
        if(root==null){
            return null;
        }

        if(root.val==val){
            return root;
        }

        TreeNode left = searchBST(root.left,val);
        TreeNode right = searchBST(root.right,val);

        //return which is not null or return null
        return left!=null?left:right;

    }
}
```

Saved
Ln 34, Col 1

Test Result

Accepted Runtime: 0 ms