## CHANDIGARH UNIVERSITY

## UNIVERSITY INSTITUTE OF NGINEERING

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



|  |  |
| --- | --- |
| **Submitted By: Submitted To:**  Vivek Kumar(21BCS8129) Mamta Punia(E12337) | |
| **Subject Name** | Competitive Coding - I |
| **Subject Code** | 20CSP-314 |
| **Branch** | Computer Science and Engineering |
| **Semester** | 5th |

**Experiment No. - 6**

**Student Name: Vivek Kumar UID: 21BCS8129**

**Branch: BE-CSE(LEET) Section/Group: WM-20BCS-616/A**

**Semester: 5th Date of Performance: 14/10/2022**

**Subject Name: Competitive coding - I Subject Code: 20CSP-314**

# **Tree: Top View**

**1. Aim/Overview of the practical:**

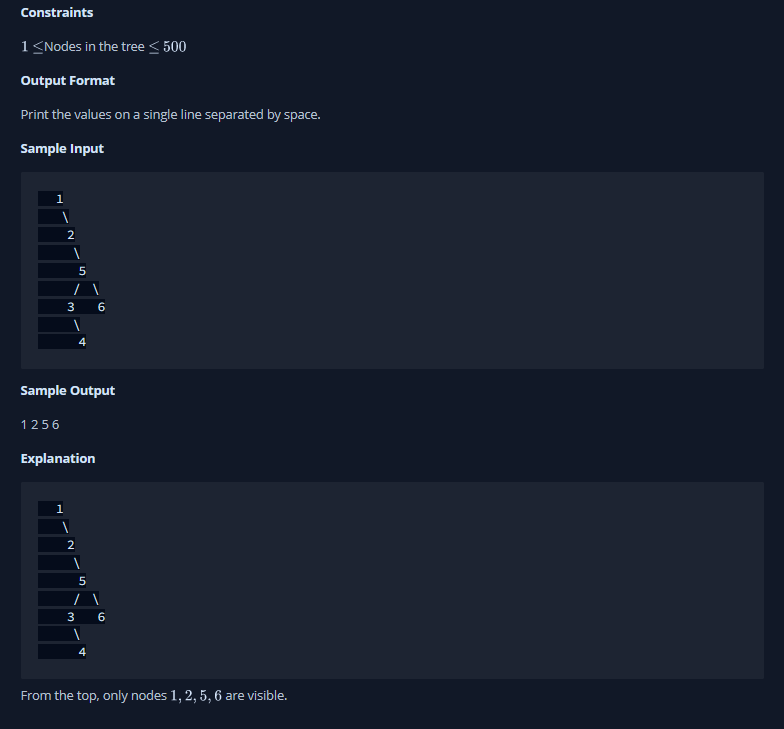
Given a pointer to the root of a binary tree, print the top view of the binary tree.

The tree as seen from the top the nodes, is called the top view of the tree.

**2. Task to be done/ Which logistics used:**







**3. Hardware and Software Requirements (For programming-based labs):**

* Laptop or Desktop
* Hacker-Rank Account

**4. Steps for experiment/practical/Code:**

static class QueueObj {

Node node;

int hd;

QueueObj(Node node, int hd)

{

this.node = node;

this.hd = hd;

}

}

public static void topView(Node root)

{

if (root == null)

return;

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

Map<Integer, Integer> map = new HashMap<>();

int min = 0;

int max = 0;

//Level Order Traversal

q.add(new QueueObj(root, 0));

while (!q.isEmpty()) {

QueueObj curr = q.poll();

//only include in map if this is the

//first node of this specific

//horizontal distance

if (!map.containsKey(curr.hd)) {

map.put(curr.hd, curr.node.data);

}

if (curr.node.left != null) {

//min can be found only in left side due to "-1"

//minimum horizontal distance of any node from root

min = Math.min(min, curr.hd - 1);

q.add(

new QueueObj(curr.node.left, curr.hd - 1));

}

if (curr.node.right != null) {

//max can be found only in right side due to "+1"

//maximum horizontal distance of any node from root

max = Math.max(max, curr.hd + 1);

q.add(

new QueueObj(curr.node.right, curr.hd + 1));

}

}

//traversal of (horizontal distance from root)

//minimum to maximum

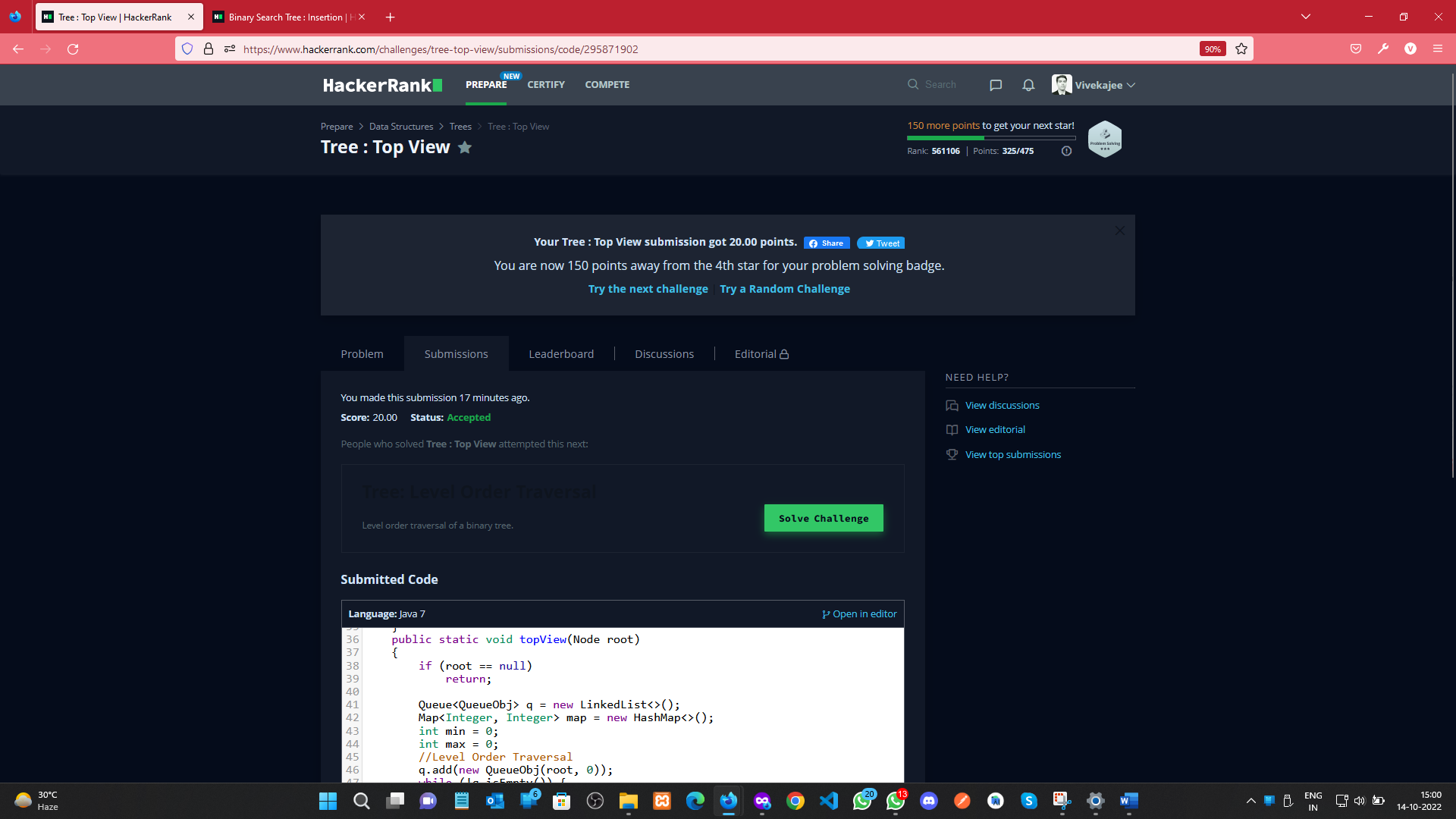
for (; min <= max; min++) {

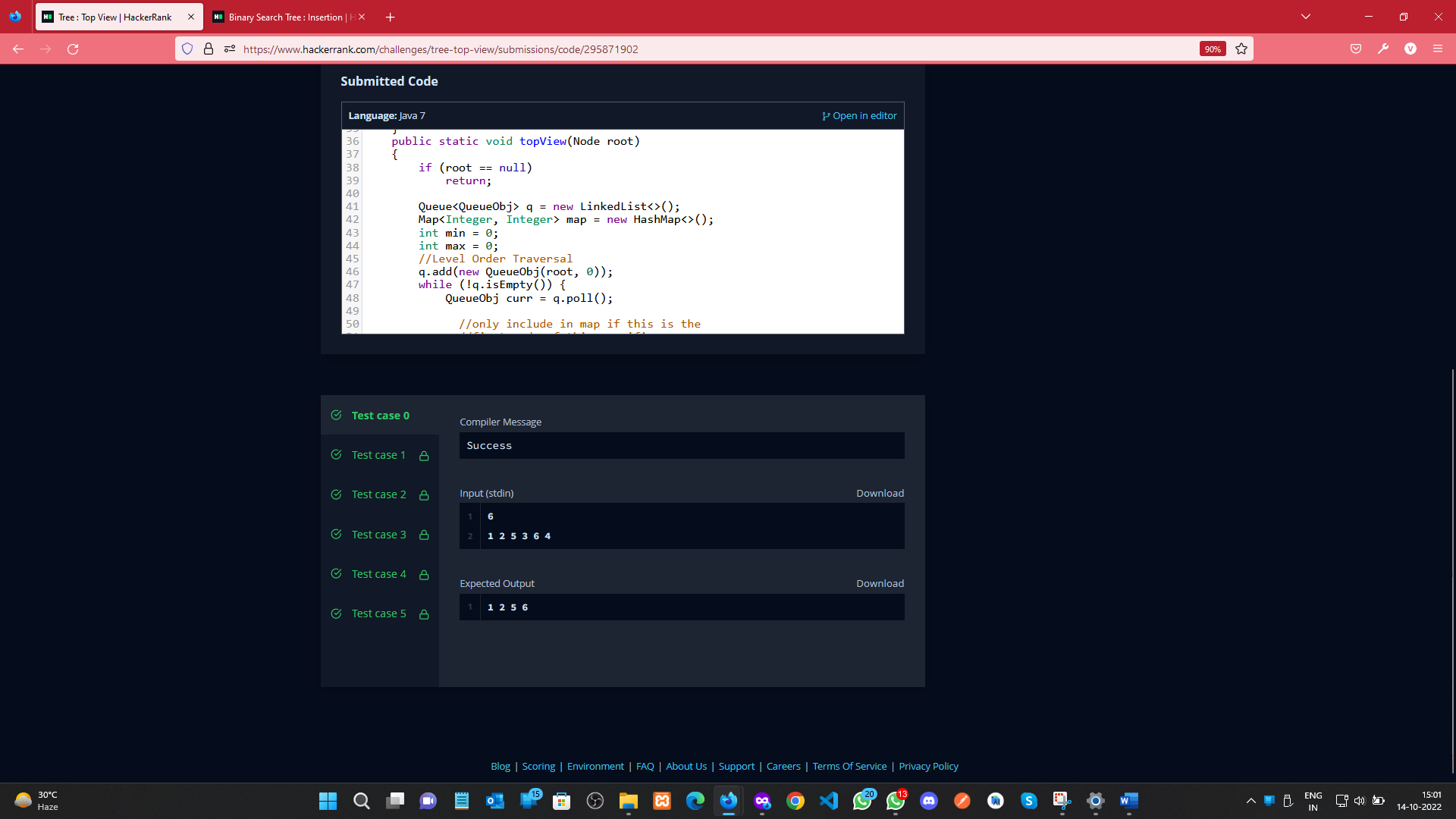
System.out.print(map.get(min)+" ");

}

}

**5. Result/Output/Writing Summary:**





**Binary Search Tree: Insertion**

**1. Aim/Overview of the practical:**

You are given a pointer to the root of a binary search tree and values to be inserted into the tree. Insert the values into their appropriate position in the binary search tree and return the root of the updated binary tree. You just have to complete the function.

**2. Task to be done/ Which logistics used:**





**3. Hardware and Software Requirements (For programming-based labs):**

* Laptop or Desktop
* Hacker-Rank Account

**4. Steps for experiment/practical/Code:**

import java.io.\*;

import java.util.\*;

class Node {

Node left;

Node right;

int data;

Node(int data) {

this.data = data;

left = null;

right = null;

}

}

public class Solution {

public static void preOrder( Node root ) {

if( root == null)

return;

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

preOrder(root.left);

preOrder(root.right);

}

public static Node insert(Node root,int value)

{

if(root == null) {

root = new Node(value);

} else if(value < root.data){

root.left = insert(root.left,value);

} else if(value > root.data) {

root.right = insert(root.right,value);

}

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);

}

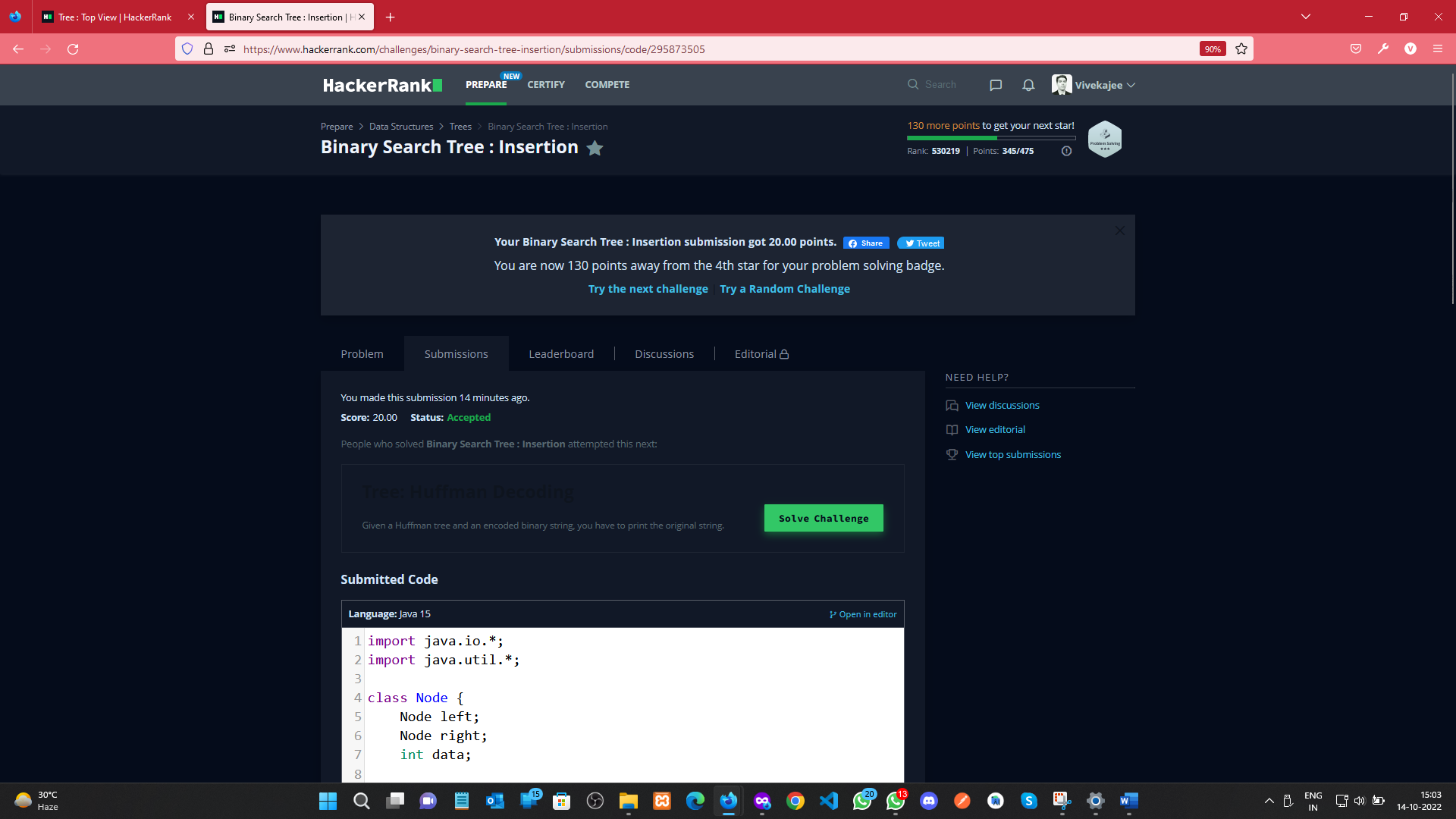
scan.close();

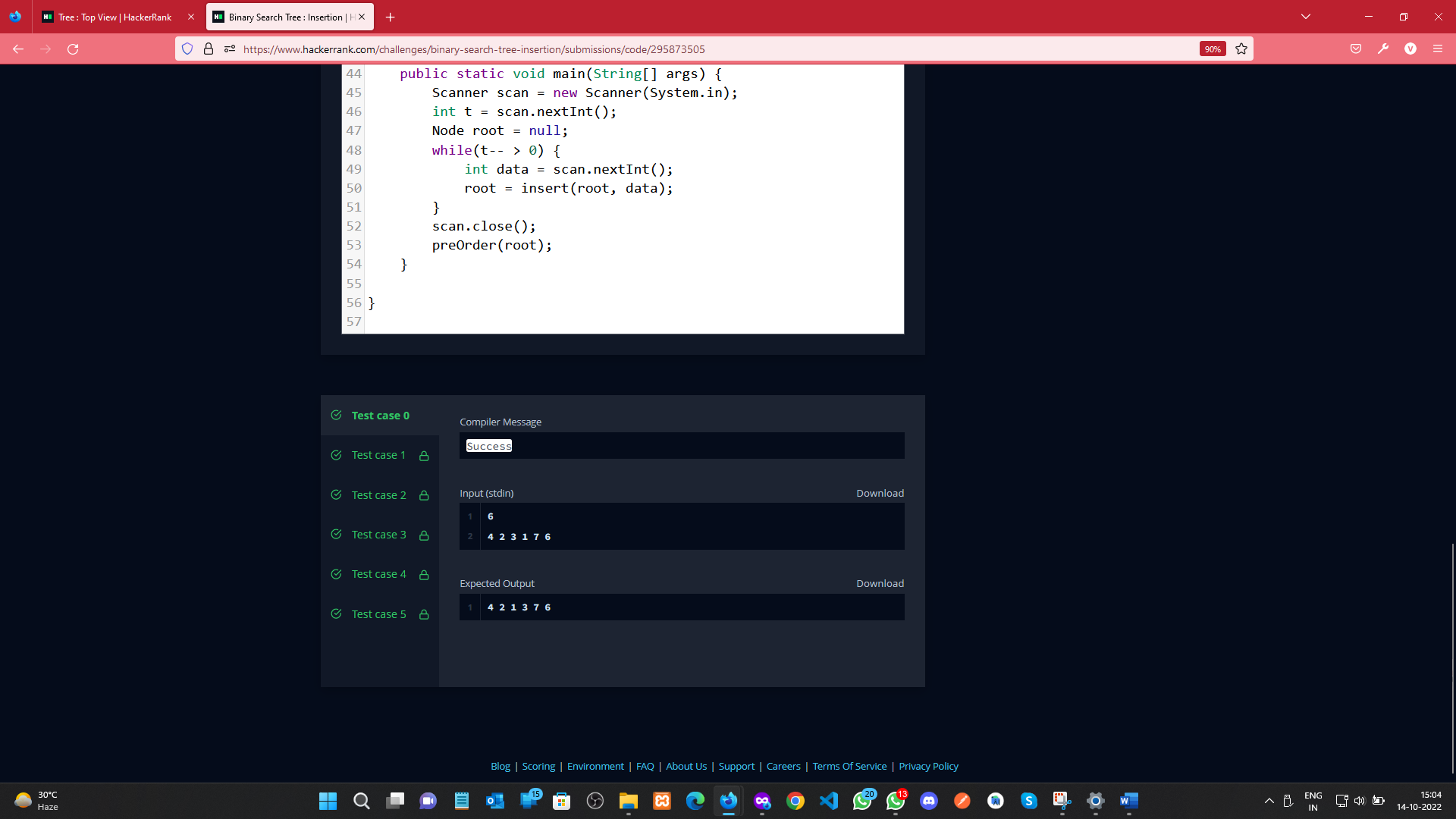
preOrder(root);

}

}

**6. Result/Output/Writing Summary:**





**Learning outcomes (What I have learnt):**

a. Learnt about Tree concept.

1. b. Learnt about Binary Search Tree.
2. c. Learn about the tree using queue.

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
|  |  |  |  |