CHANDIGARH UNIVERSITY

UNIVERSITY INSTITUTE OF NGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Submitted By: Anjali Singh(20BCS9239)	
Subject Name	Competitive Coding - I
Subject Code	20CSP-314
Branch	Computer Science and Engineering
Semester	5th

Experiment No. - 6

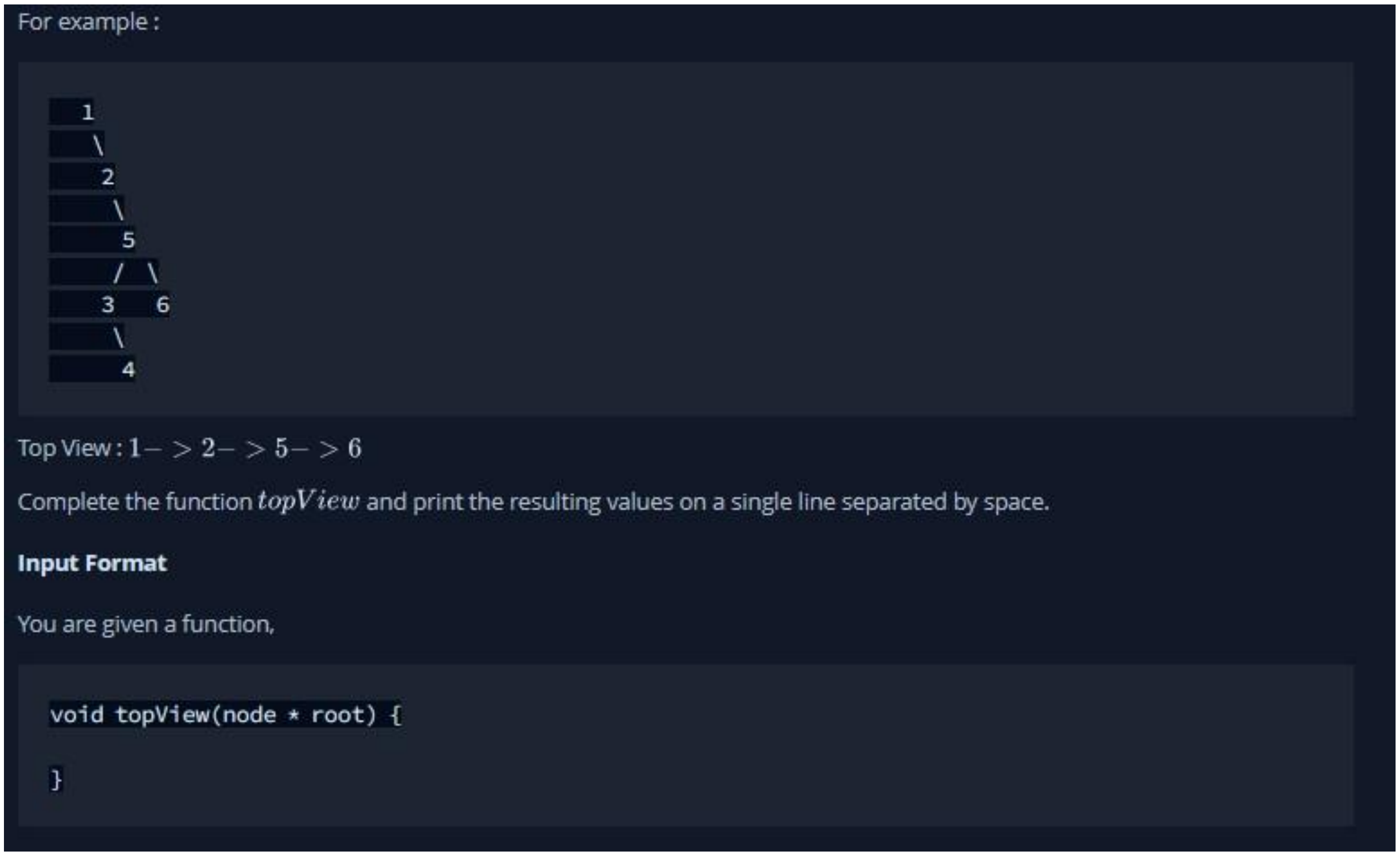
Student Name: Anjali Singh **UID: 20BCS9239**
Branch: BE-CSE **Section/Group: 607/A**
Semester: 5th **Date of Performance: 14/10/2022**
Subject Name: Competitive coding - I **Subject Code: 20CSP-314**

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:



For example :

```
   1
    \
     2
      \
       5
      /  \
     3    6
      \
       4
```

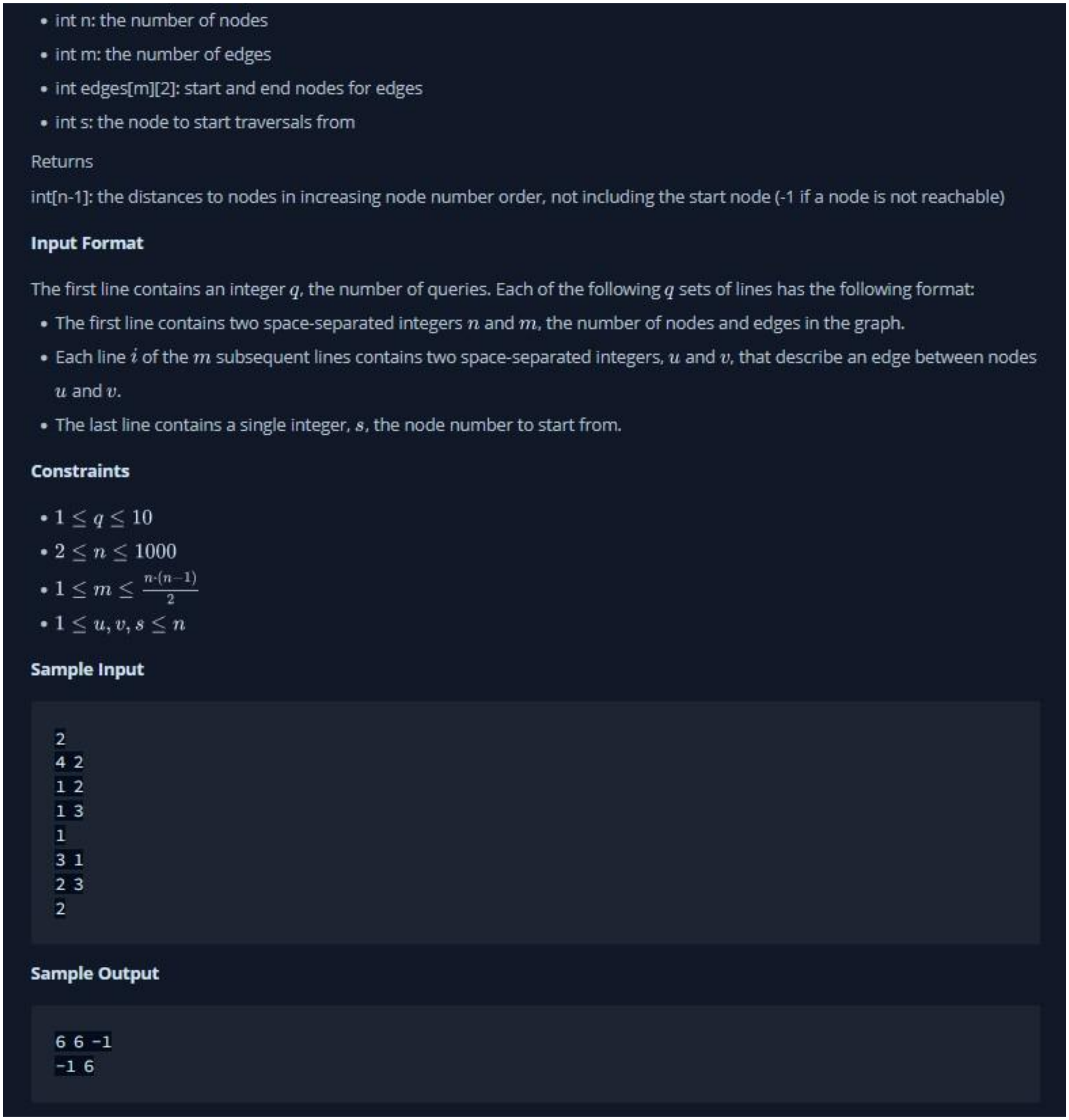
Top View : $1->2->5->6$

Complete the function $topView$ and print the resulting values on a single line separated by space.

Input Format

You are given a function,

```
void topView(node * root) {

}
```



- int n: the number of nodes
- int m: the number of edges
- int edges[m][2]: start and end nodes for edges
- int s: the node to start traversals from

Returns

int[n-1]: the distances to nodes in increasing node number order, not including the start node (-1 if a node is not reachable)

Input Format

The first line contains an integer q, the number of queries. Each of the following q sets of lines has the following format:

- The first line contains two space-separated integers n and m, the number of nodes and edges in the graph.
- Each line i of the m subsequent lines contains two space-separated integers, u and v, that describe an edge between nodes u and v.
- The last line contains a single integer, s, the node number to start from.

Constraints

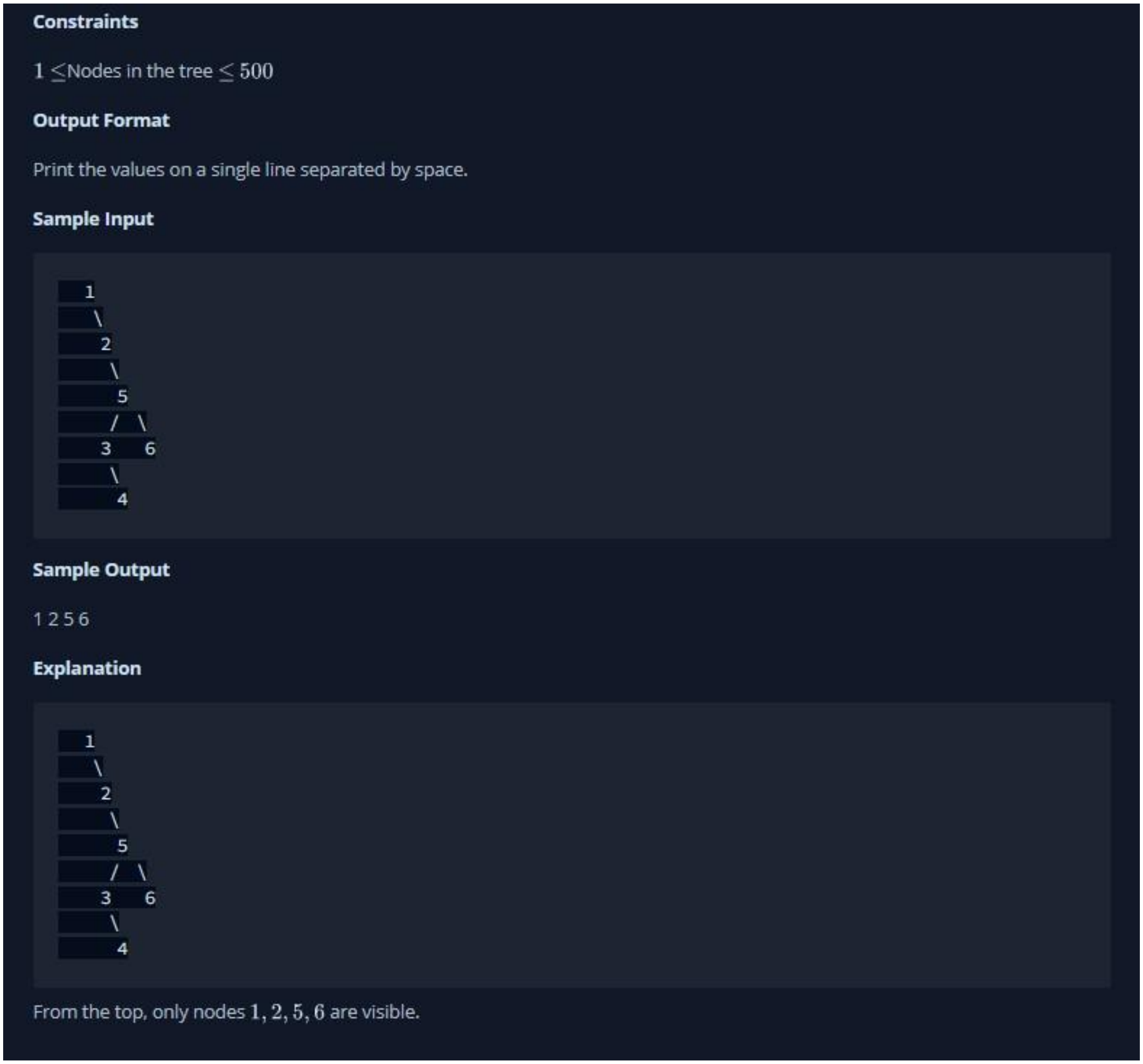
- $1 \leq q \leq 10$
- $2 \leq n \leq 1000$
- $1 \leq m \leq \frac{n \cdot (n-1)}{2}$
- $1 \leq u, v, s \leq n$

Sample Input

```
2
4 2
1 2
1 3
1
3 1
2 3
2
```

Sample Output

```
6 6 -1
-1 6
```



Constraints

$1 \leq$ Nodes in the tree ≤ 500

Output Format

Print the values on a single line separated by space.

Sample Input

```
   1
    \
     2
      \
       5
      /  \
     3    6
      \
       4
```

Sample Output

```
1 2 5 6
```

Explanation

```
   1
    \
     2
      \
       5
      /  \
     3    6
      \
       4
```

From the top, only nodes $1, 2, 5, 6$ are visible.

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

- Laptop or Desktop
- Hacker-Rank Account

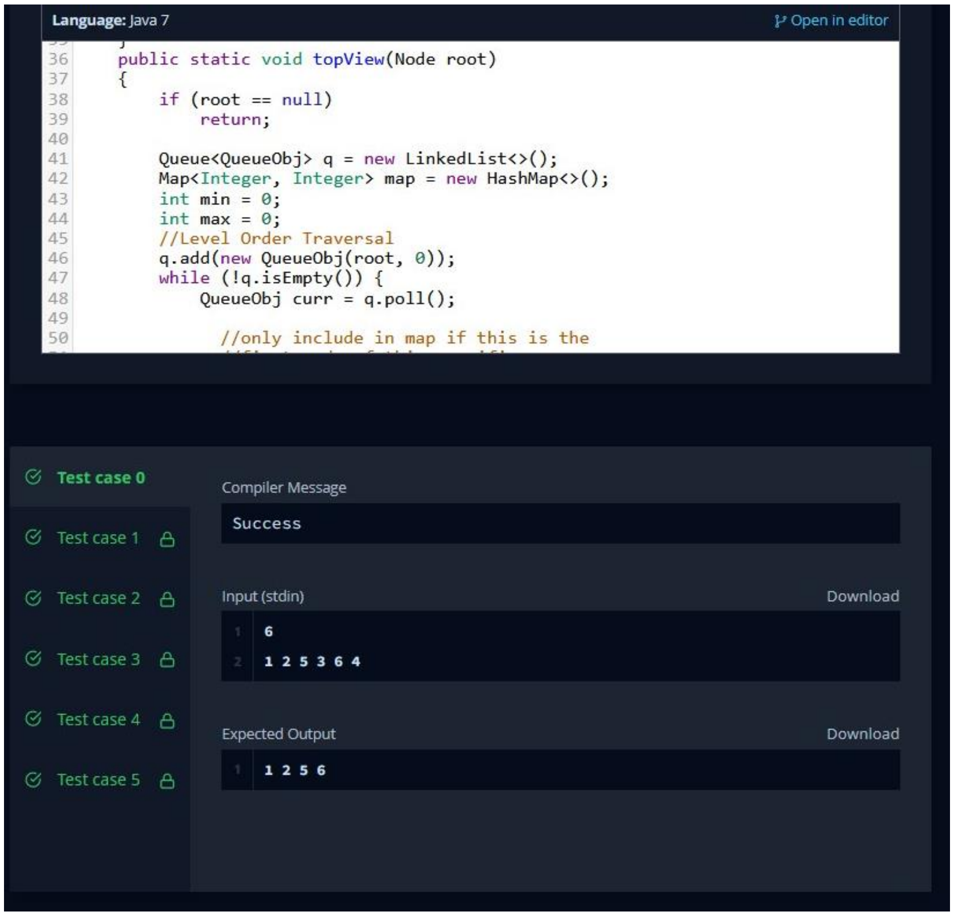
4. Steps for experiment/practical/Code:

```
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:



Language: Java 7   Open in editor

```
    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
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5

Compiler Message

```
Success
```

Input (stdin)   Download

```
6
1 2 5 3 6 4
```

Expected Output   Download

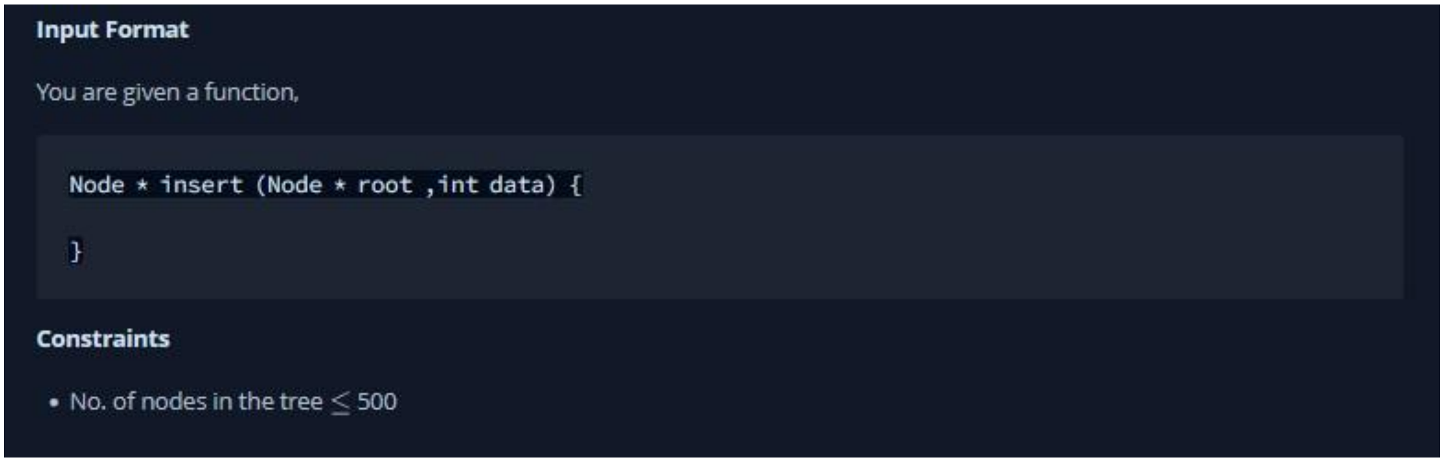
```
1 2 5 6
```

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:



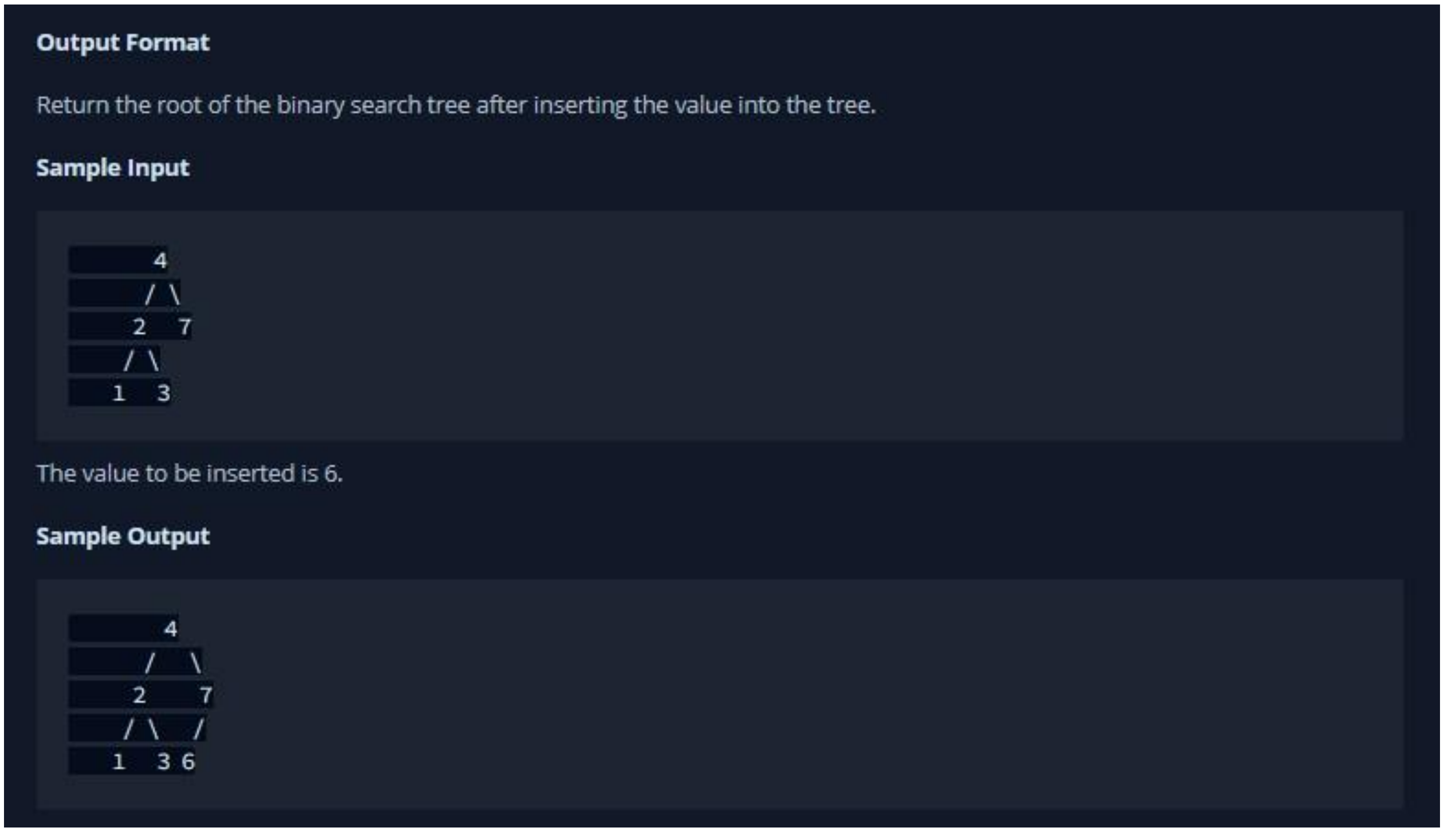
Input Format

You are given a function,

```
Node * insert (Node * root ,int data) {

}
```

Constraints

- No. of nodes in the tree ≤ 500



Output Format

Return the root of the binary search tree after inserting the value into the tree.

Sample Input

```
        4
       / \
      2   7
     / \
    1   3
```

The value to be inserted is 6.

Sample Output

```
         4
       /   \
      2     7
     / \   /
    1   3 6
```

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

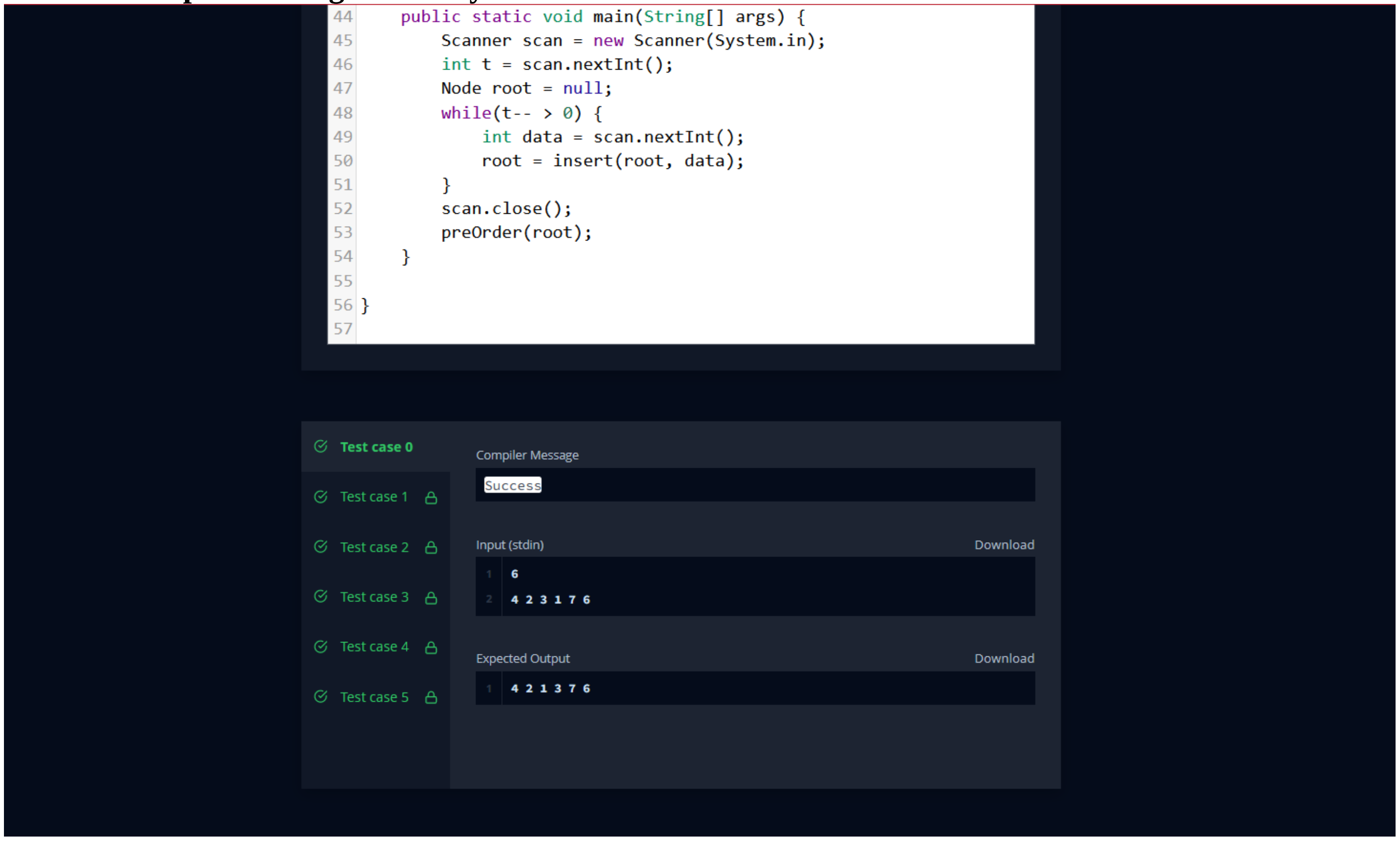
- Laptop or Desktop
- Hacker-Rank Account

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:



```
    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);
    }

}
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5

Compiler Message

```
Success
```

Input (stdin) Download

```
6
4 2 3 1 7 6
```

Expected Output Download

```
4 2 1 3 7 6
```

Learning outcomes (What I have learnt):

a. Learnt about Tree concept.

b. Learnt about Binary Search Tree.

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.			