Experiment - 6

Student Name: Jitesh Kumar **UID: 20BCS2334**
Branch: CSE **Section/Group: 20BCS-WM-903-A**
Semester: 5 **Subject Code: 20CSP-314**
Subject Name: COMPETITIVE CODING-I

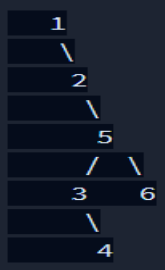
- **Aim/Overview of the practical:**
  Implementation of Hacker-rank questions.

- **Task to be done/ Which logistics used:**
  Problem Statement 1:

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.

For example :



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) {

}
```

- **Steps for experiment/practical/Code:**

```
#include<bits/stdc++.h>

using namespace std;

class Node {

    public:

        int data;

        Node *left;
```

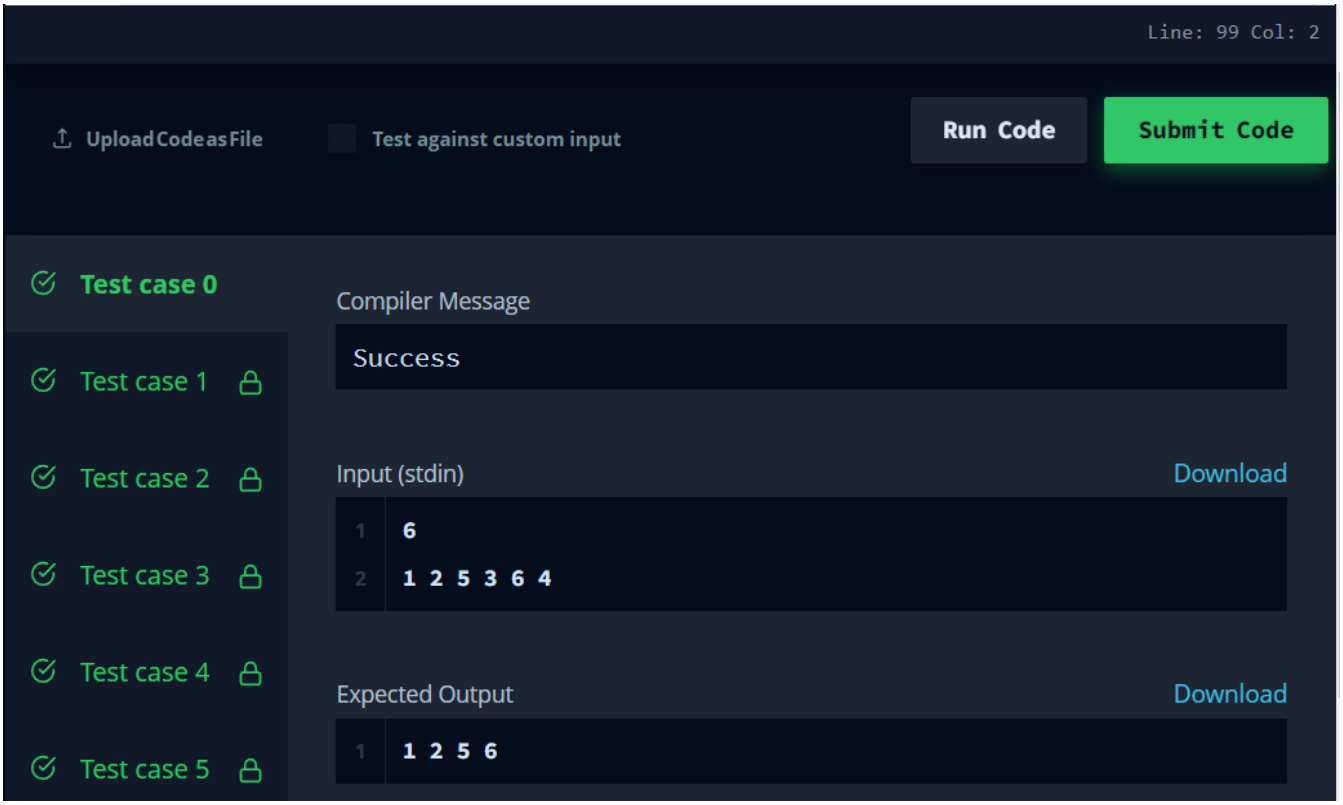
```
        Node *right;
        Node(int d) {
            data = d;
            left = NULL;
            right = NULL;
        }
};

class Solution {
    public:
        Node* insert(Node* root, int data) {
            if(root == NULL) {
                return new Node(data);
            } else {
                Node* cur;
                if(data <= root->data) {
                    cur = insert(root->left, data);
                    root->left = cur;
                } else {
                    cur = insert(root->right, data);
                    root->right = cur;
                }
```

```
                return root;
            }
        }

/*
class Node {
    public:
        int data;
        Node *left;
        Node *right;
        Node(int d) {
            data = d;
            left = NULL;
            right = NULL;
        }
};

*/

    void topView(Node * root) {
        map<int,map<int,multiset<int>>> m;
        queue<pair<Node*,pair<int,int>>> q;
        q.push(make_pair(root,make_pair(0,0)));
```

```
        while(!q.empty())
        {
            auto front = q.front();
            q.pop();
            Node* node = front.first;
            int level = front.second.first,vertical = front.second.second;
            m[vertical][level].insert(node->data);
            if(node->left!=NULL) q.push(make_pair(node->left,make_pair(level+1,vert
ical-1)));
            if(node->right!=NULL) q.push(make_pair(node->right,make_pair(level+1,ve
rtical+1)));
        }
        for(auto it: m)
        {
            for(auto jt: it.second)
            {
                for(auto kt: jt.second)
                {
                    cout<<kt<<" ";
                    // break;
                }
                    break;
            }
```

```
        }
    }

}; //End of Solution
int main() {

    Solution myTree;
    Node* root = NULL;

    int t;
    int data;

    std::cin >> t;

    while(t-- > 0) {
        std::cin >> data;
        root = myTree.insert(root, data);
    }

    myTree.topView(root);

    return 0;
}
```

OUTPUT:


Line: 99 Col: 2
Upload Code as File
Test against custom input
Run Code
Submit Code
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
1 6
2 1 2 5 3 6 4
Expected Output
Download
1 1 2 5 6

Task to be done/ Which logistics used:

Problem Statement 2:

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.

Input Format

You are given a function,

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

}
```

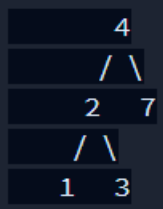
Constraints

- 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



The value to be inserted is 6.

CODE:

```
#include <iostream>
#include <cstddef>

class Node {
    public:
        int data;
        Node *left;
        Node *right;
        Node(int d) {
            data = d;
            left = NULL;
            right = NULL;
        }
};

class Solution {
    public:

    void preOrder(Node *root) {
```

```
        if( root == NULL )
            return;

        std::cout << root->data << " ";

        preOrder(root->left);
        preOrder(root->right);
    }

/* you only have to complete the function given below.
Node is defined as

class Node {
    public:
        int data;
        Node *left;
        Node *right;
        Node(int d) {
            data = d;
            left = NULL;
            right = NULL;
        }
};
```

```
*/


Node * insert(Node * root, int data) {
    Node *temp= root;
   if(root==NULL)
   {
       Node *newNode = new Node(data);
       return newNode;
   }
   else if(data<root->data)
   {
       if(root->left==NULL)
       {
           Node *newNode = new Node(data);
           root->left=newNode;
       }
       insert(root->left,data);
   }
   else if(data>root->data)
   {
```

```
            if(root->right==NULL)
            {
                Node *newNode = new Node(data);
                root->right=newNode;
            }
            insert(root->right,data);
        }
         return temp;
    }

}; //End of Solution

int main() {

    Solution myTree;
    Node* root = NULL;

    int t;
    int data;

    std::cin >> t;

    while(t-- > 0) {
```

```
            std::cin >> data;
            root = myTree.insert(root, data);
        }
        myTree.preOrder(root);
        return 0;
    }
```

OUTPUT:

Learning outcomes (What I have learnt):

1. Implementation of Tree.

2. Vertical order traversing of Tree.

3. Binary Search Tree.