Experiment- 2.2

Student Name: UTKARSH JOSHI **UID:** 21BCS9158

Branch: BE-CSE **Section/Group:** ST-802-A

Semester: 5th **Date of Performance:** 27-09-23

Subject Name: Advanced Programming Lab **Subject Code:** 21CSP-314

Aim: To implement the concept of Tree data structure.

Objective:

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.

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.

Program Code:

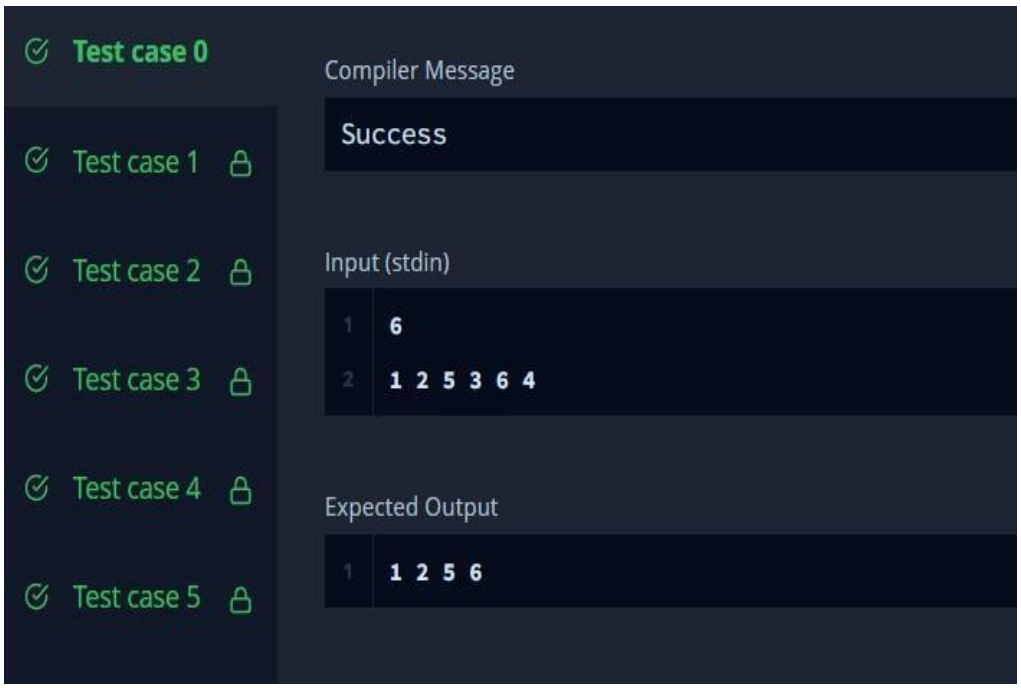
1.

```
#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) {  queue<pair<int,Node*>> q;
    q.push(make_pair(0,root)); map<int,Node*> ans;
    for(auto i=q.front();!q.empty();q.pop(),i=q.front()){
        if(!i.second) continue; ans.insert(i);
            q.push(make_pair(i.first+1,i.second->right));
        q.push(make_pair(i.first-1,i.second->left)); }
    for(auto i:ans) cout<<i.second->data<<" ";
} }; 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:



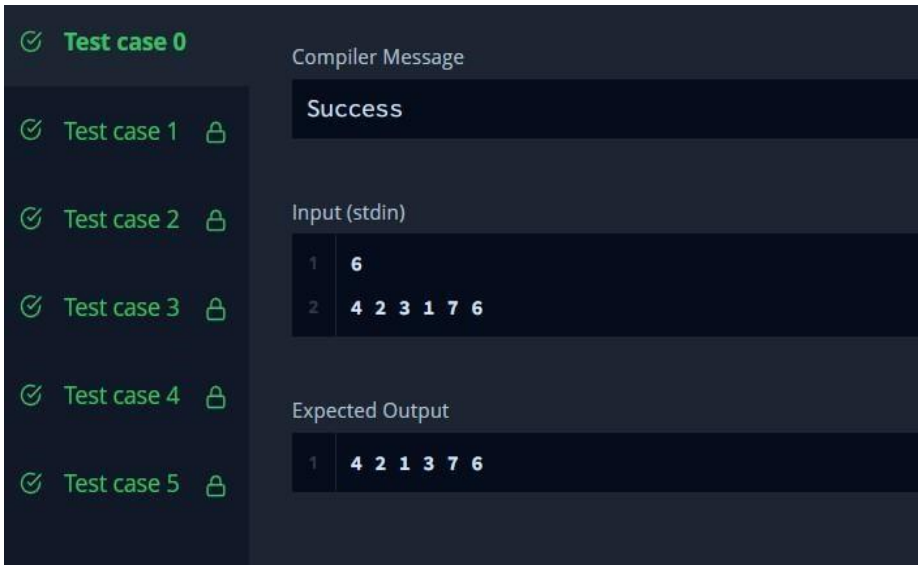
Program Code:

2.

```
#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:
    void preOrder(Node *root) {
    if( root == NULL )
```

```
            return;  std::cout << root->data << " ";
        preOrder(root->left); preOrder(root->right);
    }
    Node * insert(Node * root, int value) {
  if(root==NULL) {

    Node* newNode; newNode =
    (Node*)malloc(sizeof(Node)); newNode->left =
    NULL; newNode->right = NULL; newNode-
    >data = value; return newNode;
  } if(value <= root->data)  root->left =
    insert(root->left, value);  else  root-
    >right = insert(root->right, value);
  return root;
} }; 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:


Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
1 6
2 4 2 3 1 7 6
Expected Output
1 4 2 1 3 7 6

Learning Outcomes:

1. Learnt about Trees data structure and how to implement it.
2. Learnt about binary trees and how to insert it.
3. Learnt about Tree traversal techniques.

Experiment- 2.3

Student Name: UTKARSH JOSHI
UID: 21BCS9158
Branch: BE-CSE
Section/Group: ST-802-A
Semester: 5th
Date of Performance:27/9/23
Subject Name: Advanced Programming Lab
Subject Code: 21CSP-314

Program 1:

Aim: Determine whether a given numeric string can be split into a beautiful sequence of positive integers, adhering to specific conditions, and to find the smallest starting number if it is beautiful.

Objective: The objective of this experiment is to develop a program that assesses whether a given numeric string meets the criteria for being a "beautiful" sequence and, if so, identifies the smallest starting number of the sequence.

Program 2:

Aim: A *pangram* is a string that contains every letter of the alphabet. Given a sentence determine whether it is a pangram in the English alphabet. Ignore case. Return either pangram or not pangram as appropriate.

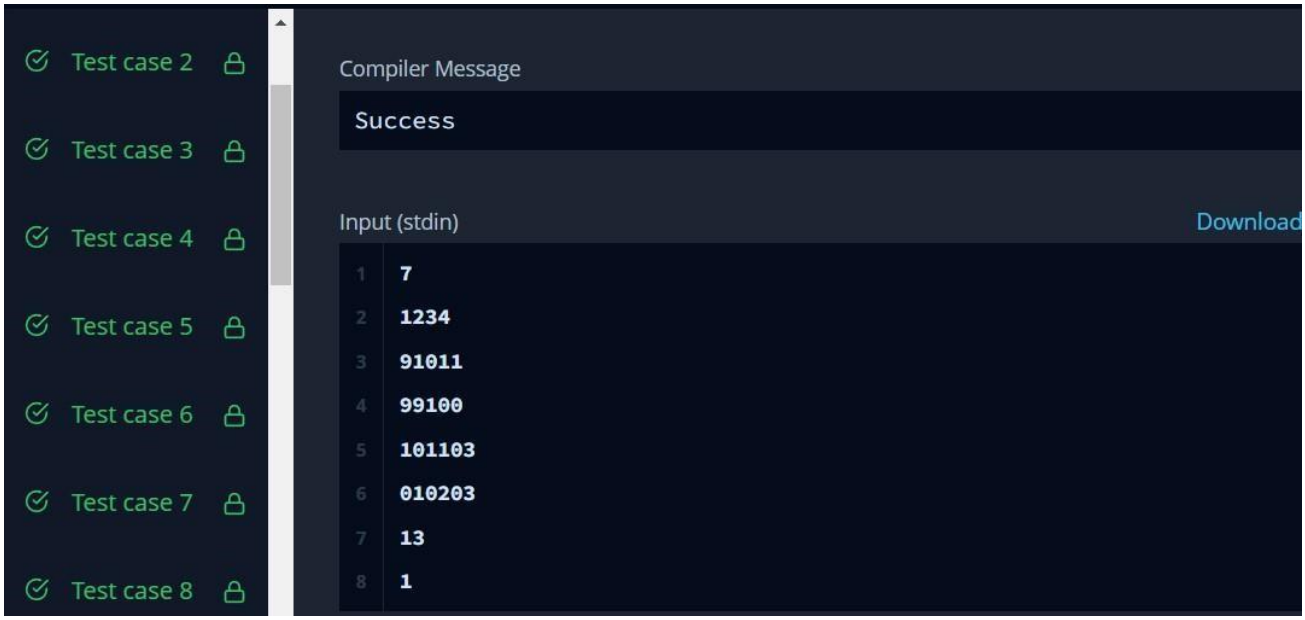
Objective: To determine whether a given sentence is a pangram, you can create a function in Python that checks if all 26 letters of the English alphabet (ignoring case) are present in the input string. You can use a set to keep track of the letters you've seen in the string and compare it to a set of all 26 English alphabet letters.

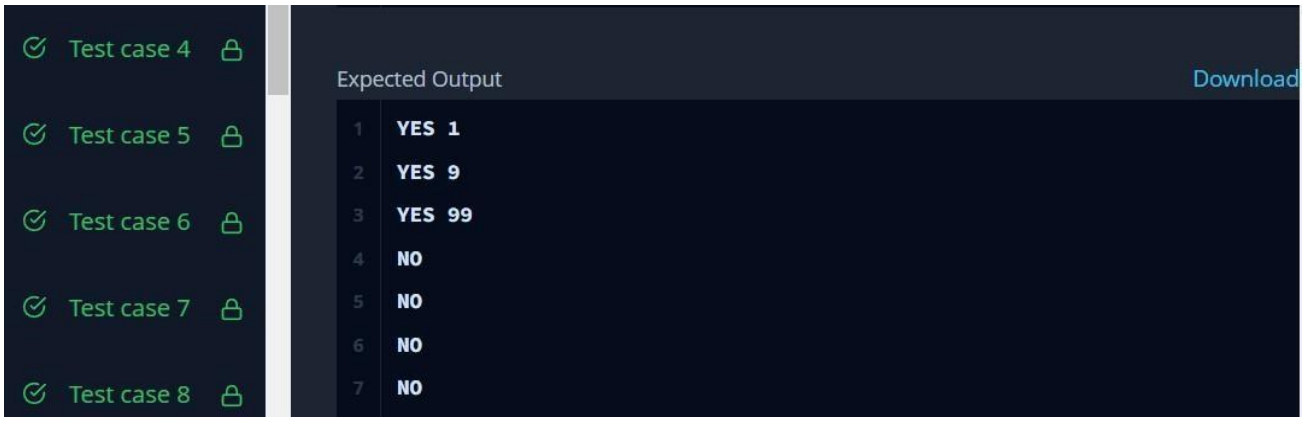
Code and Output:

Program 1:

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

using namespace std; bool isBeautiful(string s, long long
first_num) {      long long current_num = first_num;
string beautiful_sequence = to_string(current_num);

while (beautiful_sequence.length() < s.length()) {
current_num++;  beautiful_sequence +=
to_string(current_num);
}

return beautiful_sequence == s;
}

void separateNumbers(string s) { for (int len =
1; len <= s.length() / 2; len++) {      long long
first_num = stoll(s.substr(0, len));      if
(isBeautiful(s, first_num)) {         cout <<
"YES " << first_num << endl;      return;
    }
}

cout << "NO" << endl;
}

int main(){
int q;     cin
>> q;

   while (q--){         string
s; cin >> s;
separateNumbers(s);
       }        return
0;
}
```

Output:



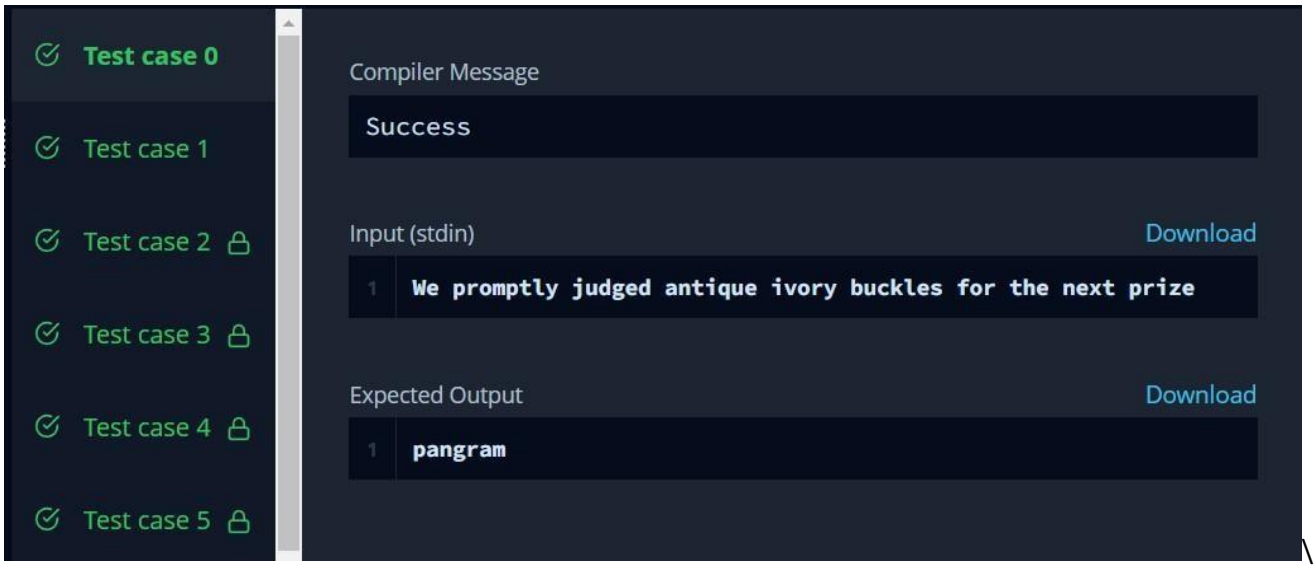


Program 2 :

```
#include <bits/stdc++.h>
using namespace std;

string check_string(string s){

    set<char> letters;
```

```
    for (char c : s) {

        if (isalpha(c)) {
letters.insert(tolower(c));
        }
    }

    if (letters.size() == 26) {
return "pangram";
        }

    else{
        return "not pangram";
        }

}

int main() {
          string s;      getline(cin, s);
string result = check_string(s);
cout << result << endl;

return 0;  }
```

Output:



Learning And Outcomes:

- Learned the implementation of String in Hacker Rank.
- Learned about pangrams and how to determine a pangram.