DSA_DAY9

Insertion , Deletion and Creation of Binary Search Tree

```
#include <bits/stdc++.h>
struct Node
{
    int val;
    Node *left;
    Node *right;
    Node(int x)
    {
        val = x;
        left = right = NULL;
    }
};

Node *insert(Node *node, int val)
{
    if (node == NULL)
        return new Node(val);
    else if (node->val == val)
        return node;
    else if (node->val < val)
        node->right = insert(node->right, val);
    else
        node->left = insert(node->left, val);
    return node;
}

Node*leftMostNode(Node* node){
    while(node->left) node=node->left;
    return node;
```

```
}

Node* Delete(Node* node,int key){
    if(node->val < key) node->right = Delete(node->right,key)
    else if(node->val > key) node->left  = Delete(node->left,
    else{
        if(node->left == nullptr) return node->right;
        else if(node->right == nullptr) return node->left;
        else{
            Node* temp  = leftMostNode(node->right);
            node->val = temp->val;
            node->right = Delete(node->right,node->val);
        }
    }
}


void inorder(Node *root)
{
    if (root == NULL)
        return ;
    inorder(root->left);
    cout << root->val << " ";
    inorder(root->right);
}

void solve()
{
    // Your solution here
    Node *root = new Node(50);
    insert(root, 10);
    insert(root, 20);
    insert(root, 30);
    insert(root, 40);
    inorder(root);
    Node* start = Delete(root,50);
    cout<<endl<<start->val<<endl;;
    inorder(root);
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.precision(10);
    int t;
    cin >> t;
    while (t--)
        solve();
    return 0;
}
```

https://www.geeksforgeeks.org/problems/check-for-bst/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=practice_card

```
class Solution {
  public:
    // Function to check whether a Binary Tree is BST or not.
    Node* prev = NULL;
    bool inorder(Node* root){
        if(!root) return true;
        if(!inorder(root->left)) return false;
        if(prev!=NULL && prev->data>=root->data) return false
        prev = root;
        if(!inorder(root->right)) return false;
        return true;

    }


    bool isBST(Node* root) {
        // Your code here
        return inorder(root);


    }
};
```

https://www.geeksforgeeks.org/problems/left-view-of-binary-tree/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=practice_card

```



class Solution {
  public:
  #define pb push_back
  vector<int> ans;
  void inorder(Node* root,int level,int& maxiLevel){
      if(!root) return ;
      if(level > maxiLevel){
          ans.pb(root->data);
          maxiLevel = level;
      }
      inorder(root->left,level+1,maxiLevel);
      inorder(root->right,level+1,maxiLevel);



  }
    vector<int> leftView(Node *root) {
        // Both Recursion and Queue Based Approach
        int maxiLevel = -1;
        inorder(root,0,maxiLevel);
        queue<Node*>q;
        q.push(root);
        int n = 1;
        if(!root) return ans;
        while(!q.empty()){
            ans.pb(q.front()->data);
            n=q.size();
            for(int i=0;i<n;i++){
                Node* node = q.front();
                if(node->left) q.push(node->left);
                if(node->right) q.push(node->right);
                q.pop();
            }
        }
        return ans;
    }
};
```

https://leetcode.com/problems/binary-tree-right-side-view/submissions/1462267020/

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val
 * };
 */
class Solution {
public:
#define pb push_back
vector<int>ans;
void order(TreeNode* root,int level,int& maxiLevel){
    if(!root) return ;
    if(level > maxiLevel) {
        ans.pb(root->val);
        maxiLevel = level;
    }
    order(root->right,level+1,maxiLevel);
    order(root->left,level+1,maxiLevel);
}
vector<int> rightSideView(TreeNode* root) {
        int maxiLevel = -1;
        order(root,0,maxiLevel);
        return ans;
    }
};
```

https://www.geeksforgeeks.org/problems/top-view-of-binary-tree/1

```
//{ Driver Code Starts
// Initial Template for C++

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

// Tree Node
struct Node {
    int data;
    Node* left;
    Node* right;
};

// Utility function to create a new Tree Node
Node* newNode(int val) {
    Node* temp = new Node;
    temp->data = val;
    temp->left = NULL;
    temp->right = NULL;

    return temp;
}

// Function to Build Tree
Node* buildTree(string str) {
    // Corner Case
    if (str.length() == 0 || str[0] == 'N')
```

```
        return NULL;

    // Creating vector of strings from input
    // string after spliting by space
    vector<string> ip;

    istringstream iss(str);
    for (string str; iss >> str;)
        ip.push_back(str);

    // Create the root of the tree
    Node* root = newNode(stoi(ip[0]));

    // Push the root to the queue
    queue<Node*> queue;
    queue.push(root);

    // Starting from the second element
    int i = 1;
    while (!queue.empty() && i < ip.size()) {

        // Get and remove the front of the queue
        Node* currNode = queue.front();
        queue.pop();

        // Get the current node's value from the string
        string currVal = ip[i];

        // If the left child is not null
        if (currVal != "N") {

            // Create the left child for the current node
            currNode->left = newNode(stoi(currVal));

            // Push it to the queue
            queue.push(currNode->left);
        }
        // For the right child
        i++;
        if (i >= ip.size())
            break;
        currVal = ip[i];

        // If the right child is not null
        if (currVal != "N") {

            // Create the right child for the current node
            currNode->right = newNode(stoi(currVal));

            // Push it to the queue
            queue.push(currNode->right);
        }
        i++;
    }

    return root;
}


// } Driver Code Ends
/*
struct Node
{
    int data;
    Node* left;
    Node* right;
};
*/
class Solution {
  public:
    // Function to return a list of nodes visible from the to
    // from left to right in Binary Tree.


    //recursion
```

```
#define pb push_back
vector<int> ans;
map<int,pair<int,int>> mp;
void order(Node* root,int h,int v){
    if(!root) return ;
    order(root->left,h-1,v+1);
    order(root->right,h+1,v+1);
    if(mp.find(h) == mp.end() || v<mp[h].second ){
        mp[h] = {root->data,v};
    }
}
vector<int> topView(Node *root) {
    // code here
    if(!root) return ans;
    // order(root,0,0);
    queue<pair<Node*,int>>q;
    q.push({root,0});
    mp[0] = {root->data,0};
    while(!q.empty()){
        int n=q.size();
        for(int i=0;i<n;i++){
            Node * node = q.front().first;
            int h = q.front().second;
            if(node->left) {
                q.push({node->left,h-1});
                if(mp.find(h-1) == mp.end()){
                        mp[h-1] = {node->left->data,h-1};
                    }
                }
            if(node->right){
                q.push({node->left,h+1});
                if(mp.find(h+1) == mp.end()){
                    mp[h+1] = {node->right->data,h+1};
                }
            }
            q.pop();
        }
        }
        for(auto i:mp){
            ans.pb(i.second.first);
        }
        return ans;

    }
};


//{ Driver Code Starts.

int main() {
    int tc;
    cin >> tc;
    cin.ignore(256, '\n');
    while (tc--) {
        string treeString;
        getline(cin, treeString);
        Solution ob;
        Node* root = buildTree(treeString);
        vector<int> vec = ob.topView(root);
        for (int x : vec)
            cout << x << " ";
        cout << endl;

        cout << "~"
             << "\n";
    }
    return 0;
}
// } Driver Code Ends
```

https://www.geeksforgeeks.org/problems/bottom-view-of-binary-tree/1

```
//Function to return a list containing the bottom view of the
```

```
class Solution {
  public:
      //recursion
    #define pb push_back
    vector<int> ans;
    map<int,pair<int,int>> mp;
    void order(Node* root,int h,int v){
        if(!root) return ;
        order(root->left,h-1,v+1);
        order(root->right,h+1,v+1);
        if(mp.find(h) == mp.end() || v>=mp[h].second ){
            mp[h] = {root->data,v};
        }
    }
    vector <int> bottomView(Node *root)
        // Your Code Here
        order(root,0,0);
          for(auto i:mp){
            ans.pb(i.second.first);
        }
        return ans;
    }
};
```

Iteration Based Segement Tree

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

using namespace std;

typedef vector<int> vi;
typedef vector<vi> vvi;
typedef pair<int, int> ii;
typedef vector<ii> vii;
typedef long long ll;
typedef vector<ll> vll;
typedef vector<vll> vvll;
typedef double ld;

void print(vector<int>& lst){
    for(int i=1; i<lst.size(); i++) cout<<lst[i]<<" ";
    cout<<"\n";
}
void update(int ind,vector<int>& tree,int n,int newValue){
    int actual_index = ind+n;
    tree[actual_index] = newValue;
    for(int i=actual_index/2; i>0; i/=2){
        tree[i] = tree[2*i] + tree[2*i+1];
    }
}
void segmentTreeCreation(){
    // Your solution here
    int n;
    cin>> n;
    vector<int> lst(n);
    for(int i=0; i<n; i++) cin>>lst[i];
    vector<int> tree(2*n,0);
    //Leaf Node intialization
    for(int i=0;i<n;i++){
        tree[i+n] =lst[i];
    }
    for(int i=n-1;i>0;i--) tree[i] = tree[2*i] + tree[2*i+1];
    cout<<"Before Updation:";
    print(tree);
    update(3,tree,n,10);
    cout<<"After Updation:";
    print(tree);
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.precision(10);
    int t;
    cin >> t;
    while (t--) segmentTreeCreation();
    return 0;
}
```