Code-

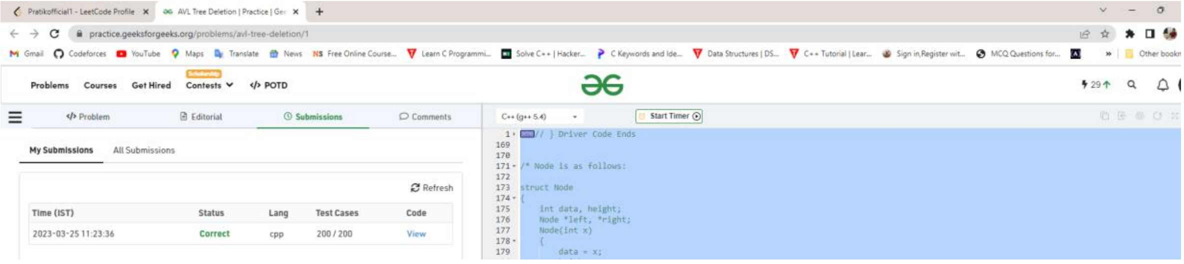
```
//{ Driver Code Starts
//

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

struct Node
{
        int data, height;
        Node *left, *right;
        Node(int x)
        {
                data = x;
                height = 1;
                left = right = NULL;
        }
};

int setHeights(Node* n)
{
        if(!n) return 0;
        n->height = 1 + max( setHeights(n->left) , setHeights(n->right) );
        return n->height;
}

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 = new Node(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 = new Node(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 = new Node(stoi(currVal));

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

    setHeights(root);
    return root;
}

bool isBST(Node *n, int lower, int upper)
{
       if(!n) return 1;
       if( n->data <= lower || n->data >= upper ) return 0;
       return isBST(n->left, lower, n->data) && isBST(n->right, n->data, upper) ;
}

pair<int,bool> isBalanced(Node* n)
{
       if(!n) return pair<int,bool> (0,1);
       pair<int,bool> l = isBalanced(n->left);
       pair<int,bool> r = isBalanced(n->right);

       if( abs(l.first - r.first) > 1 ) return pair<int,bool> (0,0);

       return pair<int,bool> ( 1 + max(l.first , r.first) , l.second && r.second
);
}

bool isBalancedBST(Node* root)
{
       if( !isBST(root, INT_MIN, INT_MAX) )
           cout<< "BST voilated, inorder traversal : ";

       else if ( ! isBalanced(root).second )
           cout<< "Unbalanced BST, inorder traversal : ";

       else return 1;
       return 0;
}

void printInorder(Node* n)
{
       if(!n) return;
       printInorder(n->left);
       cout<< n->data << " ";
       printInorder(n->right);
}

struct Node* deleteNode(struct Node* root, int data);

int main()
{
       int t;
       cin>>t;
       getchar();

       while(t--)
       {
           string s;
           getline(cin,s);
           Node* root = buildTree(s);

           int n;
           cin>> n;
           int ip[n];
           for(int i=0; i<n; i++)
               cin>> ip[i];

           for(int i=0; i<n; i++)
           {
               root = deleteNode(root, ip[i]);

               if( !isBalancedBST(root) )
                   break;
           }

           if(root==NULL)
               cout<<"null";
           else
               printInorder(root);
           cout<< endl;

           getline(cin,s); // to deal with newline char
       }
       return 1;
}

// } Driver Code Ends
```

```
/* Node is as follows:

struct Node
{
        int data, height;
        Node *left, *right;
        Node(int x)
        {
                data = x;
                height = 1;
                left = right = NULL;
        }
};

*/

int height(Node *root){
    if(!root)
        return 0;
    int leftHeight = height(root->left);
    int rightHeight = height(root->right);
    return (leftHeight>rightHeight?leftHeight:rightHeight)+1;
}
int bf(Node *root){
    if(!root)
        return 0;
    int leftHeight = height(root->left);
    int rightHeight = height(root->right);
    return rightHeight-leftHeight;
}
Node *leftRotation(Node *x){
    Node *y = x->right;
    Node *T = y->left;

    x->right = T;
    y->left = x;
    return y;
}
Node *rightRotation(Node *x){
    Node *y = x->left;
    Node *T = y->right;

    x->left = T;
    y->right = x;
    return y;
}
int findMax(Node *head){
    if(!head)
        return -1;
    while(head->left){
        head = head->left;
    }
    return head->data;
}
Node* deleteNode(Node* root, int data)
{
    //add code here,
    if(!root)
        return root;
    if(root->data<data)
        root->right = deleteNode(root->right,data);
    else if(root->data>data)
        root->left = deleteNode(root->left,data);
    else{
          if(!root->left and !root->right){
              Node *temp = root;
              root = NULL;
              delete(temp);
          }else if(!root->right){
              Node *temp = root;
              root = root->left;
              delete(temp);
          }else if(!root->left){
              Node *temp = root;
              root = root->right;
              delete(temp);
          }else{
              int maximum = findMax(root->right);
              root->data = maximum;
              root->right = deleteNode(root->right,maximum);
          }
    }
        if(!root)
            return root;
        int bff = bf(root);
          if(bff>1 and bf(root->right)>=0)
                return leftRotation(root);
          else if(bff<-1 and bf(root->left)<=0)
                return rightRotation(root);
          else if(bff>1 and bf(root->right)<0){
                root->right = rightRotation(root->right);
                return leftRotation(root);
          }
          else if(bff<-1 and bf(root->left)>0){
                root->left = leftRotation(root->left);
                return rightRotation(root);
          }
    return root;
};
```

--Time complexity –O(N!)

Space complexity –O(N^2)



Time (IST)	Status	Lang	Test Cases	Code
2023-03-25 11:23:36	Correct	cpp	200 / 200	View

```
// } Driver Code Ends


/* Node is as follows:

struct Node
{
    int data, height;
    Node *left, *right;
    Node(int x)
    {
        data = x;
```