```
void insertion(int val)
    {
        node *newnode = newnodes(val);
        if (root == 0)
        {
            root = newnode;
            return;
        }
        node *cur = root;
        node *prv = 0;
        while (cur != 0)
        {
            if (cur->val <= val)
            {
                prv = cur;
                cur = cur->right;
            }
            else
            {
                prv = cur;
                cur = cur->left;
            }
        }
        if (prv->val > val)
        {
            prv->left = newnode;
        }
        else
        {
            prv->right = newnode;
        }
    }
#include <bits/stdc++.h>
using namespace std;

class Node
{
public:
    int data;
    Node *right, *left;
    Node(int data)
    {
        this->data = data;
        this->right = 0;
        this->left = 0;
    }
};
class BST
{
public:
    Node *root;
    BST()
    {
        root = NULL;
    }
```

```
};
void add(int data, Node *&root)
{
    if (root == NULL)
    {
        Node *newnode = new Node(data);
        root = newnode;
        return;
    }
    if (data < root->data)
    {
        add(data, root->left);
    }
    else
    {
        add(data, root->right);
    }
}

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

int height(Node *root)
{
    if (root == 0)
    {
        return 0;
```

```
    }
    return 1 + max(height(root->left), height(root->right));
}
int count_Node(Node *root)
{
    if (root == 0)
    {
        return 0;
    }
    return 1 + count_Node(root->left) + count_Node(root->right);
}

int find_n(Node *root, int n)
{
    if (root == NULL)
    {
        return;
    }
    if (root->data == n)
    {
        cout << "Found" << endl;
        return;
    }
    if (root->data < n)
    {
        find_n(root->right, n);
    }
    else
    {
        find_n(root->left, n);
    }
}

void deletetion(Node *root, Node *prv, int n)
{
    if (root == NULL)
    {
        return;
    }
    if (root->data < n)
    {
        deletetion(root->right, n);
    }
    else if (root->data > n)
    {
        deletetion(root->left, n);
    }
    else
    {
        if (prv == NULL)
        {
        }
        else
        {
            if (root->left == NULL && root->right == NULL)
```

```
        {
            if (prv->left->data == n)
            {
                prv->left = NULL;
            }
            else
            {
                prv->right = NULL;
            }
            return;
        }
        else if (root->left == NULL)
        {
            if (prv->left->data == n)
            {
                prv->left = root->right;
            }
            else
            {
                prv->right = root->right;
            }
            return;
        }
        else if (root->right == NULL)
        {
            if (prv->left->data == n)
            {
                prv->left = root->left;
            }
            else
            {
                prv->right = root->left;
            }
            return;
        }
        else
        {
            Node *tm = root->right;
            while (tm->left != NULL)
            {
                tm = tm->left;
            }
            root->data = tm->data;
        }
    }
  }
}

int main()
{
   BST *bs = new BST;
   add(34, bs->root);
   add(44, bs->root);
   add(14, bs->root);
   add(4, bs->root);
```

```
    add(33, bs->root);
    add(1, bs->root);
    add(45, bs->root);
    inorder(bs->root);
}
```