```
#include <iostream>
using namespace std;
class tnode
{
      public:
      int data;
      tnode *left;
      tnode *right;
};

class bst
{
    public:
      tnode *root;
      bst()
      {
            root = NULL;
      }

      void Create()
      {
            tnode *temp = new tnode;
            temp->left = NULL;
            temp->right = NULL;
            cout << "Enter the element: ";
            cin >> temp->data;

            if (root == NULL)
            {
                  root = temp;
            }
            else
            {
                  insert(root, temp);
            }
      }

      void insert(tnode *root, tnode *newnode)
      {
            if (newnode->data < root->data)
            {
                  if (root->left == NULL)
                  {
                        root->left = newnode;
                  }
                  else
                  {
                        insert(root->left, newnode);
                  }
            }
            else if (newnode->data > root->data)
            {
                  if (root->right == NULL)
                  {
                        root->right = newnode;
                  }
                  else
```

```
            {
                insert(root->right, newnode);
            }
        }
}

void pre_display(tnode *temp)
{
    if (temp != NULL)
    {
        cout << temp->data << " ";
        pre_display(temp->left);
        pre_display(temp->right);

    }
}

void post_display(tnode *temp)
{
    if (temp != NULL)
    {
        post_display(temp->left);
        post_display(temp->right);
        cout << temp->data << " ";
    }
}

void In_display(tnode *temp)
{
    if (temp != NULL)
    {
        In_display(temp->left);
        cout << temp->data << " ";
        In_display(temp->right);
    }
}

void display()
{
    if (root == NULL)
    {
        cout << "\nTree is empty." << endl;
    }
    else
    {
        cout << "\nPreorder traversal: ";
        pre_display(root);
        cout << endl;

        cout << "\nInorder traversal: ";
        In_display(root);
        cout << endl;

        cout << "\nPostorder traversal: ";
        post_display(root);
        cout << endl;
    }
```

```
}

bool search(tnode *root, int key)
{
    if (root == NULL)
    {
        cout << "\nTree is not created." << endl;
        return false;
    }

    while (root != NULL)
    {
        if (root->data == key)
        {
            cout << "\nElement " << key << " is present." << endl;
            return true;
        }
        else if (key < root->data)
        {
            root = root->left;
        }
        else
        {
            root = root->right;
        }
    }

    cout << "\nElement " << key << " is not present." << endl;
    return false;
}

void find()
{
    int key;
    cout << "\nEnter element you want to search for: ";
    cin >> key;
    bool found = search(root, key);
}

void deleteNode(tnode *&node)
{
    if (node->left == NULL)
    {
        tnode *temp = node;
        node = node->right;
        delete temp;
    }
    else if (node->right == NULL)
    {
        tnode *temp = node;
        node = node->left;
        delete temp;
    }
    else
    {
        tnode *temp = node->right;
        while (temp->left != NULL)
```

```
        {
            temp = temp->left;
        }
        node->data = temp->data;
        deleteNode(node->right);
    }
}

void del(tnode *&root, int key)
{

    if (root == NULL)
    {
        cout << "\nTree is not created." << endl;
        return;
    }

    if (key < root->data)
    {
        del(root->left, key);
    }
    else if (key > root->data)
    {
        del(root->right, key);
    }
    else
    {
        deleteNode(root);
        cout << "\nElement " << key << " is deleted." << endl;
    }
}

int Depth(tnode* root)
{
    if (root == nullptr) {
        return 0;
    }

    int leftDepth = Depth(root->left);
    int rightDepth = Depth(root->right);

    if (leftDepth > rightDepth)
        return leftDepth + 1;
    else
        return rightDepth + 1;
}

void ParentChild (tnode* root)
{
    if (root != nullptr)
    {
        if (root->left != nullptr || root->right != nullptr)
        {
            cout << "\nParent: " << root->data<< " --> ";
            if (root->left != nullptr)
            {
                cout << "Left Child: " << root->left->data << " ";
```

```
                }
                if (root->right != nullptr)
                {
                    cout << "Right Child: " << root->right->data;
                }
                cout <<"\n";
            }
            ParentChild(root->left);
            ParentChild(root->right);
        }
    }
    void LeafNodes(tnode* root)
    {

        if (root != nullptr)
        {
            if (root->left == nullptr && root->right == nullptr)
            {
                cout << root->data << " ";
            }
            LeafNodes(root->left);
            LeafNodes(root->right);
        }
    }

};

int main()
{
    int choice;
    char ans = 'n';
    bst b;
    cout << "-------BINARY SEARCH TREE-------\n";

    do {

        cout << "\n1.Create\n2.Search\n3.Delete\n4.Display\n5.Depth\n\n";
        cout << "Enter choice: ";
        cin >> choice;
        switch (choice)
        {
            case 1:
                    do
                    {
                        b.Create();
                        cout << "Want to enter more elements? (y/n): ";
                        cin >> ans;
                    } while (ans == 'y' || ans == 'Y');
                    break;
            case 2:
                    b.find();
                    break;
            case 3:
                    int key;
                    cout << "Enter element to delete: ";
                    cin >> key;
                    b.del(b.root,key);
```

```
                    break;
            case 4:
                    b.display();
                    b.ParentChild (b.root);
                    cout<<"\nLeaf node : ";
                    b.LeafNodes(b.root);
                    break;
            case 5:
                    cout<<"Tree Depth: " << b.Depth(b.root)<<"\n";
                    break;

        }
        cout << "\nWant to continue? (y/n): ";
        cin >> ans;
        cout<<"------------------------------\n";
        } while (ans == 'y' || ans == 'Y');
    return 0;
}
```