Assignment 11

Program:

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

using namespace std;

class node

{

   public:

   int data;

   node *pLeft;

   node *pRight;

   node(int d)

   {

      data=d;

      pLeft=NULL;

      pRight=NULL;

   }

};

class BST

{

   public:

   node *pRoot;


   BST()

   {

      pRoot=NULL;

   }
```

```
};

node *Insert( node *&ppRoot, int iNo)

{

    node *pNewNode = new node(iNo);

    node *p = ppRoot;

    node *q;

    if (ppRoot == NULL)

    {

        ppRoot = pNewNode;

        return pNewNode;

    }

    while (p != NULL)

    {

        q = p;

        if (iNo < p->data)

        {

            p = p->pLeft;

        }

        else

            p = p->pRight;

    }

    if (iNo < q->data)

    {

        q->pLeft = pNewNode;

    }

    else
    {
        q->pRight = pNewNode;
    }
    return ppRoot;
}
void InOrder(node *pRoot)
{
    if (pRoot == NULL)
        return;

    InOrder(pRoot->pLeft);
    cout << "Element is :" << pRoot->data<<"\t";
    InOrder(pRoot->pRight);
}
void PreOrder(node *pRoot)
{
    if (pRoot == NULL)
    {
        return;
    }
    cout << "Element is :" << pRoot->data<<"\t";
    PreOrder(pRoot->pLeft);
    PreOrder(pRoot->pRight);
}
void PostOrder(node *pRoot)
{
    if (pRoot == NULL)
    {
        return;
    }
    PostOrder(pRoot->pLeft);
    PostOrder(pRoot->pRight);
    cout << "Element is :" << pRoot->data<<"\t";
}
node* LargestNodeBst(node* pRoot) {
    node* pCurrent = pRoot;
    while (pCurrent && pCurrent->pRight != NULL) {
        pCurrent = pCurrent->pRight;
    }
    return pCurrent;
}
node*Delete(node *ppRoot,int iNo)
{
    if (ppRoot == NULL)
        return ppRoot;

    if ((ppRoot)->data < iNo)
        (ppRoot)->pRight = Delete((ppRoot)->pRight,iNo);
    else if ((ppRoot)->data > iNo)
        (ppRoot)->pLeft = Delete((ppRoot)->pLeft,iNo);
    else {
        if ((ppRoot)->pLeft == NULL && (ppRoot)->pRight == NULL) {
            free(ppRoot);
            return NULL;
        }
        else if ((ppRoot)->pLeft == NULL) {
        node* pTemp = (ppRoot)->pRight;
            free(ppRoot);
            return pTemp;
        }
        else if ((ppRoot)->pRight == NULL) {
            node* pTemp = (ppRoot)->pLeft;
            free(ppRoot);
            return pTemp;
        }
        else {
            node* JustSmallerNode = LargestNodeBst((ppRoot)->pLeft);
            (ppRoot)->data = JustSmallerNode->data;
            (ppRoot)->pLeft = Delete((ppRoot)->pLeft, JustSmallerNode->data);
        }
    }
    return ppRoot;
}

bool searchBST(node* pRoot, int iNo) {
    if (pRoot == NULL)
        return false;
    if (pRoot->data == iNo)
        return true;

    if (pRoot->data < iNo)
        return searchBST(pRoot->pRight,iNo);

    if (pRoot->data >iNo)
        return searchBST(pRoot->pLeft,iNo);
}
int main(void)
{
    BST bst1;
    int iChoice;
    int iValue;
     do {
        cout << "Menu:" << endl;
        cout << "1. Insert" << endl;
        cout << "2. Delete" << endl;
        cout << "3. Search" << endl;
        cout << "4. Inorder Traversal" << endl;
        cout << "5. Preorder Traversal" << endl;
        cout << "6. Postorder Traversal" << endl;
        cout << "7. Exit" << endl;
        cin>>iChoice;
        switch (iChoice) {
        case 1:
```

```
    cout << "Enter value to insert: ";
    cin >> iValue;
    Insert(bst1.pRoot,iValue);
    break;

case 2:
    cout << "Enter value to delete: ";
    cin >> iValue;
    bst1.pRoot = Delete(bst1.pRoot, iValue);
    break;

case 3:
    cout << "Enter value to search: ";
    cin >> iValue;
    if (searchBST(bst1.pRoot,iValue))
        cout << "Value found in the BST." << endl;
    else
        cout << "Value not found in the BST." << endl;
    break;

case 4:
    cout << "Inorder Traversal: ";
    InOrder(bst1.pRoot);
    cout << endl;
    break;
```

```
        case 5:
          cout << "Preorder Traversal: ";
          PreOrder(bst1.pRoot);
          cout << endl;
          break;

        case 6:
          cout << "Postorder Traversal: ";
          PostOrder(bst1.pRoot);
          cout << endl;
          break;
        case 7:
          break;
        default:
          cout << "Invalid choice. Please try again." << endl;
        }

    } while (iChoice != 8);

    return 0;
}
```

Output:

```
Menu:
1. Insert
2. Delete
3. Search
```

```
4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit

1

Enter value to insert: 50

Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit

1

Enter value to insert: 30

Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit

1

Enter value to insert: 20
```

```
Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit

1

Enter value to insert: 35

Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit

1

Enter value to insert: 75

Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal
```

```
6. Postorder Traversal

7. Exit

1

Enter value to insert: 80

Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit

1

Enter value to insert: 32

Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit

5

Preorder Traversal: Element is :50      Element is :30  Element is :20  Element is :35  Element is
:32  Element is :75 Element is :80

Menu:

1. Insert
```

```
2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit

2

Enter value to delete: 75

Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit

4

Inorder Traversal: Element is :20      Element is :30  Element is :32  Element is :35  Element is
:50  Element is :80

Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit
```

```
3

Enter value to search: 20

Value found in the BST.

Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit

7

Menu:

1. Insert

2. Delete

3. Search

4. Inorder Traversal

5. Preorder Traversal

6. Postorder Traversal

7. Exit
```