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

class Node
{
public:
    int data;
    Node *left;
    Node *right;

    Node(int data)
    {
        this->data = data;
        this->left = NULL;
        this->right = NULL;
    }
};

Node *insertInBST(Node *root, int data)
{
    if (root == NULL)
    {
        root = new Node(data);
        return root;
    }
    if (root->data > data)
    {
        root->left = insertInBST(root->left, data);
    }
    else
    {
        root->right = insertInBST(root->right, data);
    }
    return root;
}

void takeInput(Node *&root)
{
    cout << "Enter Elements you want to Insert(Enter -1 to exit) : ";
    int data;
    cin >> data;
    while (data != -1)
    {
        root = insertInBST(root, data);
        cin >> data;
    }
}

bool searchInBST(Node *root, int key)
{
    if (root == NULL)
    {
        return false;
    }
    if (root->data == key)
    {
        return true;
    }
    else if (root->data > key)
    {
        // left part
        return searchInBST(root->left, key);
    }
    else
    {
        return searchInBST(root->right, key);
    }
}

Node *minEle(Node *root)
{
    Node *temp = root;
    while (temp->left != NULL)
    {

        temp = temp->left;
    }
    return temp;
}

Node *maxEle(Node *root)
{
    while (root->right != NULL)
    {
        root = root->right;
    }
    return root;
}

Node *deleteFromBST(Node *root, int key)
{
    if (root == NULL)
    {
        return root;
    }
    if (root->data == key)
    {
        // 0 child
        if (root->left == NULL && root->right == NULL)
        {
            delete root;
            return NULL;
        }
        // 1 left child
        if (root->left != NULL && root->right == NULL)
        {
            Node *temp = root->left;
            delete root;
            return temp;
        }

        // 1 right child
        if (root->left == NULL && root->right != NULL)
        {
            Node *temp = root->right;
            delete root;
            return temp;
        }

        // 2 child
        if (root->left != NULL && root->right != NULL)
        {
            int mini = minEle(root)->data;
            root->data = mini;
            root->right = deleteFromBST(root->right, mini);
            return root;
        }
    }
    else if (root->data > key)
    {
        root->left = deleteFromBST(root->left, key);
        return root;
    }
    else
    {
        root->right = deleteFromBST(root->right, key);
        return root;
    }
}

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

int main()
{

    Node *root = NULL;
    while (true)
    {
        cout << "1.Insert Elements." << endl;
        cout << "2.Search Element." << endl;
        cout << "3.Delete Element." << endl;
        cout << "4.Print Elements." << endl;
        cout << "5.Exit." << endl;
        cout << endl;

        int choice;
        cout << "Enter your choice : ";
        cin >> choice;
        cout << endl;

        switch (choice)
        {
        case 1:
            takeInput(root);
            break;
        case 2:
            cout << "Enter Element You Want to Search : ";
            int ele;
            cin >> ele;
            if (searchInBST(root, ele))
            {
                cout << "Element Found!" << endl;
            }
            else
            {
                cout << "Element not Found!" << endl;
            }
            cout << endl;
            break;
        case 3:
            cout << "Enter Element You Want to Delete : ";
            int ele2;
            cin >> ele2;
            root = deleteFromBST(root, ele2);
            cout << endl;
            break;
        case 4:
            cout << "Inorder : ";
            inorder(root);
            cout << endl;
            cout << "Preorder : ";
            preorder(root);
            cout << endl;
            cout << "Postorder : ";
            postorder(root);
            cout << endl;
            cout << endl;
            break;
        case 5:
            cout << "Exiting the Program.." << endl;
            exit(0);
        default:
            cout << "Enter Valid Input.." << endl;
        }
    }
}
```

Output :

```
1.Insert Elements.
2.Search Element.
3.Delete Element.
4.Print Elements.
5.Exit.

Enter your choice : 1

Enter Elements you want to Insert(Enter -1 to exit) : 10 8 9 7 13 11 18 -1
1.Insert Elements.
2.Search Element.
3.Delete Element.
4.Print Elements.
5.Exit.

Enter your choice : 4

Inorder : 7 8 9 10 11 13 18
Preorder : 10 8 7 9 13 11 18
Postorder : 7 9 8 11 18 13 10

1.Insert Elements.
2.Search Element.
3.Delete Element.
4.Print Elements.
5.Exit.

Enter your choice : 2

Enter Element You Want to Search : 13
Element Found!

1.Insert Elements.
2.Search Element.
3.Delete Element.
4.Print Elements.
```

```
5.Exit.

Enter your choice : 2

Enter Element You Want to Search : 23
Element not Found!

1.Insert Elements.
2.Search Element.
3.Delete Element.
4.Print Elements.
5.Exit.

Enter your choice : 3

Enter Element You Want to Delete : 13

1.Insert Elements.
2.Search Element.
3.Delete Element.
4.Print Elements.
5.Exit.

Enter your choice : 4

Inorder : 7 8 9 10 11 11 18
Preorder : 10 8 7 9 11 11 18
Postorder : 7 9 8 11 18 11 10

1.Insert Elements.
2.Search Element.
3.Delete Element.
4.Print Elements.
5.Exit.

Enter your choice : 3

Enter Element You Want to Delete : 9

1.Insert Elements.
2.Search Element.
3.Delete Element.
4.Print Elements.
5.Exit.

Enter your choice : 4

Inorder : 7 8 10 11 11 18
Preorder : 10 8 7 11 11 18
Postorder : 7 8 11 18 11 10

1.Insert Elements.
2.Search Element.
3.Delete Element.
4.Print Elements.
```

```
5.Exit.

Enter your choice : 5

Exiting the Program..
```