```
#include <iostream>

#include <cstdlib>

using namespace std;

struct tree

{

   int info;

   tree *Left, *Right;

};

tree* root;

class Binary_tree

{

public:

   Binary_tree();

   void insert1(int);

   tree* insert2(tree*, tree*);

   void Delete(int);

   void pretrav(tree*);

   void intrav(tree*);

   void posttrav(tree*);

};

Binary_tree::Binary_tree()

{

   root = NULL;

}

tree* Binary_tree::insert2(tree* temp, tree* newnode)
```

```
{
   if (temp == NULL)
  {
      temp = newnode;
   }
   else if (temp->info < newnode->info)
  {
      insert2(temp->Right, newnode);
      if (temp->Right == NULL)
         temp->Right = newnode;
   }
   else
  {
      insert2(temp->Left, newnode);
      if (temp->Left == NULL)
         temp->Left = newnode;
   }
   return temp;
}
void Binary_tree::insert1(int n)
{
   tree *temp = root, *newnode;
   newnode = new tree;
   newnode->Left = NULL;
   newnode->Right = NULL;
```

```
    newnode->info = n;

    root = insert2(temp, newnode);

}

void Binary_tree::pretrav(tree* t = root)

{

    if (root == NULL)

   {

        cout << "Nothing to display";

    }

    else if (t != NULL)

   {

        cout << t->info << " ";

        pretrav(t->Left);

        pretrav(t->Right);

    }

}

void Binary_tree::intrav(tree* t = root)

{

    if (root == NULL)

   {

        cout << "Nothing to display";

    }

    else if (t != NULL)

   {

        intrav(t->Left);
        cout << t->info << " ";
        intrav(t->Right);
    }
}
void Binary_tree::posttrav(tree* t = root)
{
    if (root == NULL)
   {
        cout << "Nothing to display";
    }
    else if (t != NULL)
   {
        posttrav(t->Left);
        posttrav(t->Right);
        cout << t->info << " ";
    }
}
void Binary_tree::Delete(int key)
{
    tree *temp = root, *parent = root, *marker;
    if (temp == NULL)
        cout << "The tree is empty" << endl;
    else
   {
        while (temp != NULL && temp->info != key) {
            parent = temp;

            if (temp->info < key)

            {

                temp = temp->Right;

            }

            else

           {

                temp = temp->Left;

            }

      }

 }

    marker = temp;

    if (temp == NULL)

        cout << "No node present";

    else if (temp == root)

       {

        if (temp->Right == NULL && temp->Left == NULL)

        {

            root = NULL;

        }

        else if (temp->Left == NULL)

       {

            root = temp->Right;

        }

        else if (temp->Right == NULL)
    {
        root = temp->Left;
    }
    else
    {
        tree* temp1;
        temp1 = temp->Right;
        while (temp1->Left != NULL)
        {
            temp = temp1;
            temp1 = temp1->Left;
        }
        if (temp1 != temp->Right)
        {
            temp->Left = temp1->Right;
            temp1->Right = root->Right;
        }
        temp1->Left = root->Left;
        root = temp1;
        }
    }
else {
    if (temp->Right == NULL && temp->Left == NULL) {
```

```
    if (parent->Right == temp)
        parent->Right = NULL;
    else
        parent->Left = NULL;
}
else if (temp->Left == NULL) {
    if (parent->Right == temp)
        parent->Right = temp->Right;
    else
        parent->Left = temp->Right;
}
else if (temp->Right == NULL) {
    if (parent->Right == temp)
        parent->Right = temp->Left;
    else
        parent->Left = temp->Left;
}
else {
    tree* temp1;
    parent = temp;
    temp1 = temp->Right;
    while (temp1->Left != NULL) {
        parent = temp1;
        temp1 = temp1->Left;
```

```
            }
            if (temp1 != temp->Right) {
                temp->Left = temp1->Right;
                temp1->Right = parent->Right;
            }
            temp1->Left = parent->Left;
            //parent = temp1;
        }
    }
    delete marker;
}
int main()
{
    Binary_tree bt;
    int choice, n, key;
    while (1)
    {
        cout << "\n\t1. Insert\n\t2. Delete\n\t3. Preorder Traversal\n\t4. Inorder Treversal\n\t5. Postorder
Traversal\n\t6. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
        case 1:
```

```
    cout << "Enter item: ";

    cin >> n;

    bt.insert1(n);

    break;

case 2:

    cout << "Enter element to delete: ";

    cin >> key;

    bt.Delete(key);

    break;

case 3:

    cout << endl;

    bt.pretrav();

    break;

case 4:

    cout << endl;

    bt.intrav();

    break;

case 5:

    cout << endl;

    bt.posttrav();

    break;

case 6:

    exit(0);

}
```

```
    }

    //return 0;

}
```