#include <iostream>

using namespace std;

class Node

{

public:

int data;

Node\* lchild;

Node\* rchild;

Node(int value) : data(value), lchild(NULL), rchild(NULL) {}

};

class BST {

public:

Node \*root;

BST() : root(NULL) {}

void insert(int value) {

root = insertRec(root, value);

}

void display()

{

inorder(root);

}

Node\* insertRec(Node \*node, int value) {

if (node == NULL)

{

return new Node(value);

}

else if(value < node->data)

{

node->lchild = insertRec(node->lchild, value);

}

else

{

node->rchild = insertRec(node->rchild, value);

}

return node;

}

void inorder(Node \*node)

{

if (node != NULL)

{

inorder(node->lchild);

cout<<node->data<<" ";

inorder(node->rchild);

}

}

int search(Node\* node, int elem)

{

if(node != NULL)

{

if(elem==node->data)

{

cout<<"element is present in the binary tree"<<endl;

return 1;

}

else if(elem>node->data)

search(node->rchild, elem);

else

search(node->lchild, elem);

}

else

{

cout<<"element is not present in the binary tree"<<endl;

return 0;

}

}

void del(Node \*node, int todel, int found)

{

if(found==0)

{

cout<<"cannot delete as element is not present in the tree\n";

}

else

{

if(todel>node->data)

{

del(node->rchild, todel, 1);

}

else if(todel<node->data)

{

del(node->lchild, todel, 1);

}

else

{

if(node->lchild==NULL && node->rchild==NULL)

{

node = NULL;

cout<<"Element was deleted successfully\n";

}

else if(node->lchild!=NULL && node->rchild==NULL)

{

node = node->lchild;

cout<<"Element with only left child deleted\n";

}

else if(node->rchild!=NULL && node->lchild==NULL)

{

node = node->rchild;

cout<<"Element with only right child deleted\n";

}

}

}

}

};

int main()

{

BST tree;

int op;

int data;

int cont = 0;

int elem;

int delelem;

int found;

cout<<"Enter 1 to insert element\nEnter 2 to display inorder traversal\nEnter 3 to search an element\nEnter 4 to delete element\n";

while(cont==0)

{

cout<<"Select operation to perform\n";

cin>>op;

switch(op)

{

case 1:

cout<<"Enter element to insert: ";

cin>>data;

tree.insert(data);

break;

case 2:

cout<<"Displaying tree\n";

tree.display();

cout<<endl;

break;

case 3:

cout<<"enter element to search: ";

cin>>elem;

found = tree.search(tree.root, elem);

break;

case 4:

cout<<"enter element to delete: ";

cin>>delelem;

int f;

f = tree.search(tree.root, delelem);

tree.del(tree.root, delelem, f);

break;

}

cout<<"Enter 0 to continue, 1 to exit: ";

cin>>cont;

}

return 0;

}