#include <iostream>

using namespace std;

class BST{

// declare a node of BST

struct node{

int data;

node \*left, \*right;

};

// declare the root of the BST

node\* root;

node\* insert(node\* rootNode, int element){

//If the BST is empty

if(rootNode == NULL){

rootNode = new node;

rootNode->data = element;

rootNode->left = rootNode->right = NULL;

}

//If element to insert is greater than root's data

else if(element > rootNode->data){

rootNode->right = insert(rootNode->right, element);

}

else if(element < rootNode->data){

rootNode->left = insert(rootNode->left, element);

}

else{

cout<<"Duplicate element in BST = "<< element <<endl;

}

return rootNode;

}

node\* makeEmpty(node\* rootNode) {

if(rootNode == NULL)

return NULL;

{

makeEmpty(rootNode->left);

makeEmpty(rootNode->right);

delete rootNode;

}

return NULL;

}

void inorder(node\* rootNode) {

if(rootNode == NULL)

return;

inorder(rootNode->left);

cout << rootNode->data << " ";

inorder(rootNode->right);

}

node\* findMin(node\* t)

{

if(t == NULL)

return NULL;

else if(t->left == NULL)

return t;

else

return findMin(t->left);

}

node\* findMax(node\* t) {

if(t == NULL)

return NULL;

else if(t->right == NULL)

return t;

else

return findMax(t->right);

}

node\* remove(node\* rootNode, int element) {

node\* temp;

if(rootNode == NULL){

cout<<"--NULL--"<<endl;

return NULL;

}

else if(element < rootNode->data){

cout<<"Go Left"<<endl;

rootNode->left = remove(rootNode->left, element);

}

else if(element > rootNode->data){

cout<<"Go right"<<endl;

rootNode->right = remove(rootNode->right, element);

}

else if(rootNode->left && rootNode->right) // Found the node and node has both children

{ cout<<"Two child"<<endl;

temp = findMin(rootNode->right);

rootNode->data = temp->data;

rootNode->right = remove(rootNode->right, rootNode->data);

}

else // Found the node and node has one child

{

cout<<"Delete child : "<<rootNode->data<<" - "<< rootNode->left <<" - " <<rootNode->right<<endl;

temp = rootNode;

if(rootNode->left == NULL){

cout<<"rootNode->left == NULL"<<endl;

rootNode = rootNode->right;

}

else if(rootNode->right == NULL){

cout<<"rootNode->right == NULL"<<endl;

rootNode = rootNode->left;

}

delete temp;

}

return rootNode;

}

node\* find(node\* rootNode, int x) {

if(rootNode == NULL)

return NULL;

else if(x < rootNode->data)

return find(rootNode->left, x);

else if(x > rootNode->data)

return find(rootNode->right, x);

else

return rootNode;

}

public:

BST() {

root = NULL;

}

~BST() {

root = makeEmpty(root);

}

void insert(int x) {

root = insert(root, x);

}

void display() {

inorder(root);

cout << endl;

}

void remove(int x) {

root = remove(root, x);

}

/\*void search(int x) {

node \*temp;

temp = find(root, x);

if(temp->data == x)

cout<<"found element "<<x<<endl;

else

cout<<"Element not found!!! "<<x<<endl;

}\*/

};

int main()

{

BST t;

t.insert(20);

t.insert(25);

t.insert(15);

t.insert(10);

t.insert(30);

t.insert(5);

t.insert(21);

t.insert(28);

t.insert(31);

t.display();

t.remove(25);

t.display();

/\*t.search(15);

t.search(25);\*/

return 0;

}