#include <iostream>

using namespace std;

class BST {

BinaryTreeNode<int>\* root;

public:

BST() {

root = NULL;

}

~BST() {

delete root;

}

private:

BinaryTreeNode<int>\* deleteData(int data, BinaryTreeNode<int>\* node) {

if (node == NULL) {

return NULL;

}

if (data > node->data) {

node->right = deleteData(data, node->right);

return node;

} else if (data < node->data) {

node->left = deleteData(data, node->left);

return node;

} else {

if (node->left == NULL && node->right == NULL) {

delete node;

return NULL;

} else if (node->left == NULL) {

BinaryTreeNode<int>\* temp = node->right;

node->right = NULL;

delete node;

return temp;

} else if (node->right == NULL) {

BinaryTreeNode<int>\* temp = node->left;

node->left = NULL;

delete node;

return temp;

} else {

BinaryTreeNode<int>\* minNode = node->right;

while (minNode->left != NULL) {

minNode = minNode->left;

}

int rightMin = minNode->data;

node->data = rightMin;

node->right = deleteData(rightMin, node->right);

return node;

}

}

}

void printTree(BinaryTreeNode<int>\* root) {

if (root == NULL) {

return;

}

cout << root->data << ":";

if (root->left != NULL) {

cout << "L" << root->left->data;

}

if (root->right != NULL) {

cout << "R" << root->right->data;

}

cout << endl;

printTree(root->left);

printTree(root->right);

}

public:

void deleteData(int data) {

root = deleteData(data, root);

}

void printTree() {

printTree(root);

}

private:

BinaryTreeNode<int>\* insert(int data, BinaryTreeNode<int>\* node) {

if (node == NULL) {

BinaryTreeNode<int>\* newNode = new BinaryTreeNode<int>(data);

return newNode;

}

if (data < node->data) {

node->left = insert(data, node->left);

} else {

node->right = insert(data, node->right);

}

return node;

}

public:

void insert(int data) {

this->root = insert(data, this->root);

}

private:

bool hasData(int data, BinaryTreeNode<int>\* node) {

if (node == NULL) {

return false;

}

if (node->data == data) {

return true;

} else if (data < node->data) {

return hasData(data, node->left);

} else {

return hasData(data, node->right);

}

}

public:

bool hasData(int data) {

return hasData(data, root);

}

};