#include <stdio.h>

#include <stdlib.h>

typedef struct nodo{

int valor;

struct nodo \*izdo;

struct nodo \*drcho;

struct nodo \*padre;

} Nodo;

typedef Nodo arbol;

Nodo \*CrearNodo(int valor, Nodo \*padre){

Nodo \*nuevoNodo = (Nodo \*)malloc(sizeof(Nodo));

nuevoNodo->valor = valor;

nuevoNodo->izdo = nuevoNodo->drcho = NULL;

nuevoNodo->padre = padre;

return nuevoNodo;

}

void DestruirNodo(Nodo \*nodo){

nodo->izdo = nodo->drcho = NULL;

free(nodo);

}

static void InsertarConPadre(Nodo \*\*arbol, Nodo \*padre, int valor){

if (\*arbol == NULL){

\*arbol = CrearNodo(valor, padre);

}

else{

int valorRaiz = (\*arbol)->valor;

if (valor < valorRaiz){

InsertarConPadre(&(\*arbol)->izdo, \*arbol, valor);

}else{

InsertarConPadre(&(\*arbol)->drcho, \*arbol, valor);

}

}

}

void Insertar(Nodo \*\*arbol, int valor){

InsertarConPadre(arbol, NULL, valor);

}

int Existe(Nodo \*arbol, int valor){

if (arbol == NULL){

return 0;

}else if (arbol->valor == valor){

return 1;

}else if (valor < arbol->valor){

return Existe(arbol->izdo, valor);

}else{

return Existe(arbol->drcho, valor);

}

}

Nodo \*Obtener(Nodo \*arbol, int valor){

if (arbol == NULL){

return NULL;

}else if (arbol->valor == valor){

return arbol;

}else if (valor < arbol->valor){

return Obtener(arbol->izdo, valor);

}else{

return Obtener(arbol->drcho, valor);

}

}

static void Reemplazar(Nodo \*arbol, Nodo \*nuevoNodo){

if (arbol->padre){

if (arbol->valor == arbol->padre->izdo->valor){

arbol->padre->izdo = nuevoNodo;

}else if (arbol->valor == arbol->padre->drcho->valor){

arbol->padre->drcho = nuevoNodo;

}

if (nuevoNodo){

nuevoNodo->padre = arbol->padre;

}

}

}

static Nodo \*Minimo(Nodo \*arbol){

if (arbol == NULL){

return NULL;

}if (arbol->izdo){

return Minimo(arbol->izdo);

}else{

return arbol;

}

}

static void EliminarNodo(Nodo \*nodoEliminar){

if (nodoEliminar->izdo && nodoEliminar->drcho){

//Tratar elimiar con 2 hijos

Nodo \*minimo = Minimo(nodoEliminar->drcho);

nodoEliminar->valor = minimo->valor;

EliminarNodo(minimo);

}else if (nodoEliminar->izdo){

//Tratar eliminar con 1 hijo

Reemplazar(nodoEliminar, nodoEliminar->izdo);

DestruirNodo(nodoEliminar);

}else if (nodoEliminar->drcho){

//Tratar eliminar con 1 hijo

Reemplazar(nodoEliminar, nodoEliminar->drcho);

DestruirNodo(nodoEliminar);

}else{

//Tratar eliminar con 0 hijos

Reemplazar(nodoEliminar, NULL);

DestruirNodo(nodoEliminar);

}

}

void Eliminar(Nodo \*arbol, int valor){

if (arbol == NULL){

return;

}else if (valor < arbol->valor){

Eliminar(arbol->izdo, valor);

}else if (valor > arbol->valor){

Eliminar(arbol->drcho, valor);

}else{

EliminarNodo(arbol);

}

}

void Preorden(Nodo \*arbol){

if (arbol == NULL){

printf(" - ");

}else{

printf("( %d ", arbol->valor);

Preorden(arbol->izdo);

Preorden(arbol->drcho);

printf(" )");

}

}

void Inorden(Nodo \*arbol){

if (arbol == NULL){

printf(" - ");

}else{

printf("( ");

Inorden(arbol->izdo);

printf(" %d ", arbol->valor);

Inorden(arbol->drcho);

printf(" )");

}

}

void Postorden(Nodo \*arbol){

if (arbol == NULL){

printf(" - ");

}else{

printf("( ");

Postorden(arbol->izdo);

Postorden(arbol->drcho);

printf(" %d )", arbol->valor);

}

}

int main(void){

Nodo \*arbol = NULL;

Insertar(&arbol, 5); Inorden(arbol); printf("\n");

Insertar(&arbol, 10); Inorden(arbol); printf("\n");

Insertar(&arbol, 4); Inorden(arbol); printf("\n");

Insertar(&arbol, 9); Inorden(arbol); printf("\n");

Insertar(&arbol, 15); Inorden(arbol); printf("\n");

Insertar(&arbol, 20); Inorden(arbol); printf("\n");

Insertar(&arbol, 2); Inorden(arbol); printf("\n");

Eliminar(arbol,4); Inorden(arbol); printf("\n");

Eliminar(arbol,15); Inorden(arbol); printf("\n");

Eliminar(arbol,5); Inorden(arbol); printf("\n");

}