#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

typedef struct NoLista \*PtrNoLista;

typedef struct NoLista

{

int elemento;

PtrNoLista proximo;

} NoLista;

typedef struct

{

PtrNoLista inicio;

int quant;

} ListaOrdenada;

// IMPRIMIR

// ESTA VAZIA

// TAMANHO/

// HW

// MAXIMO

// MINIMO

// PROXIMO

// ANTERIOR

// INICIAR

void iniciaLista(ListaOrdenada \*lista)

{

lista->inicio = NULL;

lista->quant = 0;

}

/////////VAZIA

bool estaVazia(ListaOrdenada \*l)

{

return (l->inicio == NULL);

}

// TAMANHO

int tamanhoLista(ListaOrdenada \*l)

{

return (l->quant);

}

// INSERIR

void inserirLista(ListaOrdenada \*lista, int x) // NÃO INSERE

{

PtrNoLista novo;

novo = malloc(sizeof(NoLista));

novo->elemento = x;

if (estaVazia(lista) || novo->elemento < lista->inicio->elemento) // PORQUE O NÃO &

{

novo->proximo = lista->inicio;

lista->inicio = novo;

}

else

{

PtrNoLista aux = lista->inicio;

while (aux->proximo != NULL && aux->proximo->elemento <= x)

{

aux = aux->proximo;

}

novo->proximo = aux->proximo;

aux->proximo = novo;

}

lista->quant++;

}

// IMPRIMIR

void imprimir(ListaOrdenada \*l)

{

printf("\n[");

PtrNoLista temp;

for (temp = l->inicio; temp != NULL; temp = temp->proximo)

{

printf("%d ", temp->elemento);

}

printf("]\n");

}

// PESQUISA burra

bool pesquisaListaBurra(ListaOrdenada \*lista, int valor)

{

PtrNoLista aux;

for (aux = lista->inicio; aux != NULL; aux = aux->proximo)

{

if (aux->elemento == valor)

{

return true;

}

}

return false;

}

// PESQUISA inteligente

bool pesquisaListaInteligente(ListaOrdenada \*lista, int valor)

{

PtrNoLista aux;

for (aux = lista->inicio; aux != NULL; aux = aux->proximo)

{

if (aux->elemento == valor)

{

return true;

}

else

{

if (aux->elemento > valor)

{

return false;

}

}

}

return false;

}

// REMOVER ELEM

bool removerLista(ListaOrdenada \*lista, int valor)

{

if (estaVazia(lista) || valor < lista->inicio->elemento) // 1 e 2

{

return false;

}

else

{

if (valor == lista->inicio->elemento) // 3

{

PtrNoLista aux = lista->inicio;

lista->inicio = lista->inicio->proximo;

free(aux);

lista->quant--;

return true;

}

else

{

PtrNoLista temp = lista->inicio;

while (temp->proximo->elemento < valor && temp->proximo != NULL)

{

temp = temp->proximo;

}

if (temp->proximo->elemento > valor || temp->proximo == NULL)

{

return false;

}

PtrNoLista remove = temp->proximo;

temp->proximo = temp->proximo->proximo;

lista->quant--;

free(remove);

return true;

}

}

}

// DESTRUIR/

int primeiroelementoLista(ListaOrdenada \*lista)

{

if (!estaVazia(lista))

{

return (lista->inicio->elemento);

}

else

{

return -9999999;

}

}

void destruirLista(ListaOrdenada \*lista)

{

int cons;

while (!estaVazia(lista))

{

cons = primeiroelementoLista(lista);

removerLista(lista, cons);

}

}

//----------------------------------------------

int main(int argc, char const \*argv[])

{

int search[] = {1, 4, 110};

int insert[] = {1, -3, 4, 5, 6, 7, 99};

int remove[] = {1, -3, 5, 6, 4, 7, 44};

ListaOrdenada lista;

iniciaLista(&lista);

if (estaVazia(&lista))

{

printf("Esta vazia ");

}

imprimir(&lista);

printf("Tamanho %d\n", tamanhoLista(&lista));

for (size\_t i = 0; i < 7; i++)

{

inserirLista(&lista, insert[i]);

}

imprimir(&lista);

for (size\_t i = 0; i < 3; i++)

{

if (pesquisaListaInteligente(&lista, search[i]))

{

printf("%d encontrado\n", search[i]);

}

else

{

printf("%d Nao encontrado\n", search[i]);

}

}

for (size\_t i = 0; i < 7; i++)

{

if (removerLista(&lista, remove[i]))

{

printf("%d removido\n", remove[i]);

}

else

{

printf("%d Nao existe\n", remove[i]);

}

}

imprimir(&lista);

destruirLista(&lista);

printf("tudo ok :)");

return 0;

}