#include <stdio.h>

#include <stdlib.h>

struct Nod{

struct Nod\* pred;

int info;

struct Nod\* succ;

};

struct LDI{

struct Nod\* front;

struct Nod\* end;

};

struct LDI\* initializare\_lista(){

struct LDI\* lista=(struct LDI\*)malloc(sizeof(struct LDI));

lista->front=NULL;

lista->end=NULL;

return lista;

}

struct Nod\* init(int valoare){

struct Nod\* nod\_nou=(struct Nod\*)malloc(sizeof(struct Nod));

nod\_nou->pred=NULL;

nod\_nou->succ=NULL;

nod\_nou->info=valoare;

return nod\_nou;

}

struct LDI\* inserare\_inceput(struct LDI\* lista, int valoare){

if(lista->front==NULL){

lista->front=init(valoare);

lista->end=lista->front;

}

else{

struct Nod\* nod\_nou=init(valoare);

nod\_nou->succ=lista->front;

lista->front->pred=nod\_nou;

lista->front=nod\_nou;

}

return lista;

}

struct LDI\* inserare\_sfarsit(struct LDI\* lista, int valoare){

if(lista->front==NULL){

lista->front=init(valoare);

lista->end=lista->front;

}

else{

struct Nod\* nod\_nou=init(valoare);

nod\_nou->pred=lista->end;

lista->end->succ=nod\_nou;

nod\_nou->succ=NULL;

lista->end=nod\_nou;

}

return lista;

}

struct LDI\* inserare\_interior(struct LDI\* lista, int valoare){

if(lista->front==NULL){

lista->front=init(valoare);

lista->end=lista->front;

}

else{

struct Nod\* nod\_nou=init(valoare);

nod\_nou->succ=lista->front->succ;

lista->front->succ->pred=nod\_nou;

nod\_nou->pred=lista->front;

lista->front->succ=nod\_nou;

}

return lista;

}

struct LDI\* stergere\_inceput(struct LDI\* lista){

if(lista->front!=NULL){

lista->front=lista->front->succ;

lista->front->pred=NULL;

}

return lista;

}

struct LDI\* stergere\_sfarsit(struct LDI\* lista){

if(lista->front!=NULL){

lista->end=lista->end->pred;

lista->end->succ=NULL;

}

return lista;

}

/\*struct LDI\* stergere\_interior(struct LDI\* lista){

if(lista->front!=NULL){

lista->front->succ->info=lista->front->info;

lista->front=lista->front->succ;

lista->front->pred=NULL;

}

return lista;

}\*/

struct LDI\* stergere\_interior(struct LDI\* lista, struct Nod\* dupa){

if(lista->front!=NULL){

struct Nod\* salv=dupa->succ;

dupa->succ=dupa->succ->succ;

salv->succ->pred=dupa;

free(salv);

}

return lista;

}

struct Nod\* cautare(struct LDI\* lista, int cheie){

struct Nod\* tmp=lista->front;

while(tmp!=NULL){

if(tmp->info==cheie){

return tmp;

}

else{

tmp=tmp->succ;

}

}

return NULL;

}

struct LDI\* actualizare(struct LDI\* lista, int valoare){

struct Nod\* adr=cautare(lista, valoare);

scanf("%d", &adr->info);

return lista;

}

int numarare(struct LDI\* lista){

struct Nod\* tmp=lista->front;

int i=0;

while(tmp!=NULL){

tmp=tmp->succ;

i++;

}

return i;

}

struct LDI\* inserare(struct LDI\* lista, int max\_num){

while(numarare(lista)<max\_num){

int valoare;

scanf("%d", &valoare);

lista=inserare\_sfarsit(lista, valoare);

}

return lista;

}

void crescator(struct LDI\* lista){

struct Nod\* tmp=lista->front;

while(tmp!=NULL){

struct Nod\* tmp1=lista->front;

while(tmp1!=NULL){

if(tmp->info<tmp1->info){

int temp=tmp->info;

tmp->info=tmp1->info;

tmp1->info=temp;

}

tmp1=tmp1->succ;

}

tmp=tmp->succ;

}

}

void descrescator(struct LDI\* lista){

struct Nod\* tmp=lista->front;

while(tmp!=NULL){

struct Nod\* tmp1=lista->front;

while(tmp1!=NULL){

if(tmp->info>tmp1->info){

int temp=tmp->info;

tmp->info=tmp1->info;

tmp1->info=temp;

}

tmp1=tmp1->succ;

}

tmp=tmp->succ;

}

}

void citire(struct LDI\* lista, int index){

struct Nod\* tmp=lista->front;

int i=0;

while(tmp!=NULL){

if(i==index){

printf("Indroduceti valoarea\n")

scanf("%d", &tmp->info);

}

i++;

tmp=tmp->succ;

}

}

void swap(struct LDI\* lista){

int temp=lista->front->info;

lista->front->info=lista->end->info;

lista->end->info=temp;

}

int main()

{

struct LDI\* lista= initializare\_lista();

lista=inserare\_inceput(lista, 3);

lista=inserare\_sfarsit(lista, 5);

lista=inserare\_interior(lista, 4);

struct Nod\* tmp=lista->front;

while(tmp!=NULL){

printf("%d\n", tmp->info);

tmp=tmp->succ;

}

// lista=stergere\_inceput(lista);

// lista=stergere\_sfarsit(lista);

// struct Nod\* dupa=lista->front;

// lista=stergere\_interior(lista, dupa);

/\* printf("%d\n", lista->front->info);

printf("%d\n", lista->front->succ->info);

printf("%d\n", lista->end->info);\*/

lista=actualizare(lista, 4);

tmp=lista->front;

while(tmp!=NULL){

printf("%d\n", tmp->info);

tmp=tmp->succ;

}

printf("Numarul de elemente este:%d\n", numarare(lista));

int max\_num;

printf("Alege nr maxim de elemente:");

scanf("%d", &max\_num);

lista=inserare(lista, max\_num);

printf("Noua lista\n");

tmp=lista->front;

while(tmp!=NULL){

printf("%d\n", tmp->info);

tmp=tmp->succ;

}

printf("Lista ordonata crescator:\n");

crescator(lista);

tmp=lista->front;

while(tmp!=NULL){

printf("%d\n", tmp->info);

tmp=tmp->succ;

}

printf("Lista ordonata descrescator:\n");

descrescator(lista);

tmp=lista->front;

while(tmp!=NULL){

printf("%d\n", tmp->info);

tmp=tmp->succ;

}

printf("Scrie indexul elementului pe care vrei sa-l schimbi\n");

int index;

scanf("%d", &index);

citire(lista, index);

printf("Lista dupa operatia de citire cu ajutorul unui index\n");

tmp=lista->front;

while(tmp!=NULL){

printf("%d\n", tmp->info);

tmp=tmp->succ;

}

swap(lista);

printf("Lista dupa ce am schimbat primul element cu ultimul\n");

tmp=lista->front;

while(tmp!=NULL){

printf("%d\n", tmp->info);

tmp=tmp->succ;

}

return 0;

}