#include <iostream>

#include <malloc>

#include <stdlib>

#define MEMCHECK(x) if(x==NULL) { cout<<"Nema memorija!\n"; exit(-1); }

typedef struct element

{

int info;

struct element \*sleden;

} clen;

main()

{

clen \*p,\*q,\*r;

clen \*mk\_link\_list(int);

void printlist (clen \*);

void free\_link\_list(clen \*);

clen \*join(clen \*, clen \*);

int n;

cout<<"Kolku clena: ";

cin>>n;

p=mk\_link\_list(n); /\* vnesuvanje na prvata lista \*/

cout<<"Kolku clena: ";

cin>>n;

q=mk\_link\_list(n); /\* vnesuvanje na vtorata lista \*/

printlist(p); /\* pecatenje na prvata lista \*/

printlist(q); /\* pecatenje na vtorata lista \*/

r=join(p,q); /\* soedinuvanje na listsite \*/

printlist(r); /\* pecatenje na izleznata lista \*/

cout<<"\n\n";

free\_link\_list(r);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* clen \*mk\_link\_list(int n) Vnesuvanje na linearna lista \*

\* \*

\* Se kreira linearna povrzana lista so n elementi so \*

\* dinamicka alokacija na memorija za sekoj jazel. Potoa od \*

\* tastatura se citaat vrednosti koi se smestuvaat vo info \*

\* polinjata na jazlite. \*

\* \*

\* Vlezni parametri: \*

\* int n - broj na elementi vo listata sto treba da se kreira \*

\* Izlezni parametri: \*

\* clen \* - funkcijata vraka pokazuvac kon pocetokot na \*

\* kreiranata lista \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

clen \*mk\_link\_list(int n)

{

clen \*p=NULL,\*q=NULL;

void fillel(clen \*);

while(n--)

{

p=(clen \*)malloc(sizeof(clen));

MEMCHECK(p);

p->sleden=q;

q=p;

}

while(q!=NULL)

{

fillel(q);

q=q->sleden;

}

return(p);

}

void fillel(clen \*q)

{

cin>>q->info;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* void printlist (clen \*p) Pecatenje na linearna lista \*

\* \*

\* Se pecati linearna povrzana lista na ekran. \*

\* \*

\* Vlezni parametri: \*

\* clen \*p - pokazuvac kon pocetokot na listata \*

\* Izlezni parametri: nema \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void printlist (clen \*p)

{

void printclen(clen \*);

while(p!=NULL)

{

printclen(p);

p=p->sleden;

}

cout<<"\n";

}

void printclen(clen \*p)

{

cout<<p->info;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* void free\_link\_list(clen \*p) Brise linearna lista \*

\* \*

\* Ja brise linearnata lista od memorija. Se osloboduva \*

\* memorijata zafatena od listata kon cij pocetok pokazuva p. \*

\* \*

\* Vlezni parametri: \*

\* clen \*p - pokazuvac kon pocetokot na listata \*

\* Izlezni parametri: nema \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void free\_link\_list(clen \*p)

{

clen \*q;

while(p!=NULL)

{

p=(q=p)->sleden;

free(q);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* clen \*join(clen \*p, clen \*q) Spojuva dve sortirani listi \*

\* \*

\* Elementite od dvete sortitani vo ratecki redosled listi \*

\* kon koi pokazuvaat p i q gi kombinira vo edna lista pri \*

\* sto se zadrzuva sortiranosta t.e. i izleznata lista e \*

\* sortirana vo rastecki redosled. \*

\* \*

\* Vlezni parametri: \*

\* clen \*p - pokazuvac kon pocetokot na prvata listata \*

\* clen \*q - pokazuvac kon pocetokot na vtorata listata \*

\* Izlezni parametri: \*

\* clen \* funkcijata vraka pokazuvac kon izleznata lista \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

clen \*join(clen \*p, clen \*q)

{

clen \*r,\*s;

if(p->info < q->info)

{

s=r=p;

p=p->sleden;

}

else

{

s=r=q;

q=q->sleden;

}

while(p!=NULL && q!=NULL)

{

if(p->info < q->info)

{

r->sleden=p;

r=p;

p=p->sleden;

}

else

{

r->sleden=q;

r=q;

q=q->sleden;

}

}

if(p!=NULL)

r->sleden=p;

else

r->sleden=q;

return(s);

}