//LLEncSimples

typedef struct SNODO NODO;

typedef float INFO;

struct SNODO {

INFO info;

NODO \* prox;

};

void CriarLLEncSimples(NODO \*\*LL) {

\*LL = NULL;

}

void DestruirLLEncSimples(NODO \*\*LL) {

NODO \* aux;

aux = \*LL;

while (\*LL != NULL) {

\*LL = (\*LL)->prox;

delete aux;

aux = \*LL;

}

}

void InserirInicioLLEncSimples(NODO \*\*LL, INFO info) {

NODO \* aux;

aux = new NODO;

aux->info = info;

aux->prox = \*LL;

\*LL = aux;

}

void InserirInicioLLEncSimples2(NODO \*\*Head, NODO \*\*Tail, INFO info) {

NODO \* aux;

aux = new NODO;

aux->info = info;

aux->prox = \*Head;

if (Head == NULL) {

\*Tail = aux;

}

\*Head = aux;

}

void InserirFinalLLEncSimples1(NODO \*\*LL, INFO info) {

NODO \* aux;

NODO \* tail;

tail = \*LL;

while (tail != NULL && tail->prox != NULL) {

tail = tail->prox;

}

aux = new NODO;

aux->info = info;

aux->prox = NULL;

if (tail == NULL) {

\*LL = aux;

tail = aux;

} else {

tail->prox = aux;

}

}

void InserirFinalLLEncSimples2(NODO \*\*Head, NODO \*\*Tail, INFO info) {

NODO \* aux = new NODO;

aux->info = info;

aux->prox = NULL;

if (\*Tail == NULL) {

\*Head = aux;

\*Tail = aux;

} else {

(\*Tail)->prox = aux;

\*Tail = aux;

}

}

NODO \* RemoverInicioLLEncSimples1(NODO \*\* LL) {

NODO \* aux = \*LL;

if (aux != NULL) {

\*LL = aux->prox;

aux->prox = NULL;

}

return (aux);

}

NODO \* RemoverInicioLLEncSimples2(NODO \* Head, NODO \* Tail) {

NODO \* aux = Head;

if (aux != NULL) {

Head = aux->prox;

aux->prox - NULL;

if (Head == NULL) {

\*Tail = \*Head;

}

}

return (aux);

}

NODO \* ObterEnderecoPenultimoNoLLEncSimples(NODO \* LL) {

if (LL == NULL)

return LL;

if (LL->prox == NULL)

return LL->prox;

if ((LL->prox)->prox == NULL)

return LL;

return ObterEnderecoPenultimoNoLLEncSimples(LL->prox);

}

NODO \* RemoverFinalLLEncSimples(NODO \*\* LL){

NODO \*ult,\*pnult;

pnult = ObterEnderecoPenultimoNoLLEncSimples(\*LL);

ult = \*LL;

if(pnult != NULL){

ult = pnult->prox;

pnult->prox = NULL;

}else{

if(\*LL != NULL){

\*LL = NULL;

}

}

return ult;}

//LLEncDup

#include <iostream>

using namespace std;

typedef int INFO;

typedef struct SDNODO NODO;

struct SDNODO{

INFO info;

NODO \* prox;

NODO \* ant;

};

void InicializarLLEncDup(NODO \*\* LLDup){

\*LLDup = NULL;

}

void MostrarLLEncSimples(NODO \*LLDup) {

while (LLDup != NULL) {

cout << LLDup->info << "->";

LLDup = LLDup->prox;

}

cout << "/" << endl;

}

void DestruirLLEncDup(NODO \*\* LLDup){

NODO \* aux;

aux = \*LLDup;

while (\*LLDup != NULL) {

\*LLDup = (\*LLDup)->prox;

delete aux;

aux = \*LLDup;

}

}

void InserirInicioLLEncDup(NODO \*\* LLDup, INFO info){

NODO \* aux;

aux = new NODO;

aux->ant = NULL;

aux->info = info;

aux->prox = \*LLDup;

if(\*LLDup != NULL){

(\*LLDup)->ant = aux;

}

\*LLDup = aux;

}

NODO\* RemoverInicioLLEncDup(NODO \*\* LLDup){

NODO \* aux;

aux = \*LLDup;

if(LLDup == NULL){

return \*LLDup;

}

\*LLDup = (\*LLDup)->prox;

if(\*LLDup != NULL){

(\*LLDup)->ant = NULL;

}

aux->prox = NULL;

return aux;

}

NODO \* ObterUltimoNodoLLEndDup(NODO \* LLEncDup){

NODO \* aux;

aux = LLEncDup;

while(aux != NULL && aux->prox != NULL){

aux = aux->prox;

}

return aux;

}

void InserirFinalLLEncDup(NODO \*\* LLEncDup, INFO info){

NODO \* aux;

NODO \* aux2;

aux = ObterUltimoNodoLLEndDup(\*LLEncDup);

aux2 = new NODO;

aux2->info = info;

aux2->prox = NULL;

aux2->ant = aux;

if(aux != NULL){

aux->prox = aux2;

}else{

\*LLEncDup = aux2;

}

}

NODO \* RemoverFinalLLEncDup(NODO \*\* LLEncDup){

NODO \*ult,\*pnult;

ult = ObterUltimoNodoLLEndDup(\*LLEncDup);

if(ult != NULL){

pnult = ult->ant;

if(pnult != NULL){

pnult->prox = NULL;

}else{

\*LLEncDup = NULL;

}

ult->ant = NULL;

}else{

\*LLEncDup = NULL;

}

return ult;

}

void OrdenarLLEncDup(NODO \*LL) {

NODO \*LL2;

LL2 = LL;

while (LL != NULL) {

LL2 = LL->prox;

while(LL2 != NULL){

if(LL->info > LL2->info){

INFO aux = LL->info;

LL->info = LL2->info;

LL2->info = aux;

}

LL2 = LL2->prox;

}

LL = LL->prox;

}

}

//PILHA

typedef DNODO LLEncDup;

typedef LLEncDup Pilha;

void InicializarPilha(Pilha \*\* pilha){

InicializarLLEncDup(pilha);

}

void DestruirPilha(Pilha \*\* pilha){

DestruirLLEncDup(pilha);

}

void Push(Pilha \*\* pilha, INFO info){

InserirFinalLLEncDup(pilha, info);

}

NODO \* Pop(Pilha \*\* pilha){

RemoverFinalLLEncDup(pilha);

}

INFO Get(Pilha \* pilha){

NODO \* aux;

INFO info;

aux = Pop(&pilha);

if(aux != NULL){

info = aux->info;

Push(&pilha,info);

}

return info;

}

//FILA

typedef DNODO LLEncDup;

typedef LLEncDup Fila;

void InicializarFila(Fila \*\* fila){

InicializarLLEncDup(fila);

}

void DestruirFila(Fila \*\* fila){

DestruirLLEncDup(fila);

}

void Enqueue(Fila \*\* fila, INFO info){

InserirFinalLLEncDup(fila, info);

}

NODO \* Dequeue(Fila \*\* fila){

RemoverInicioLLEncDup(fila);

}

INFO Get(Fila \* fila){

NODO \* aux;

INFO info;

aux = Dequeue(&fila);//tira ele do comeÃ§o

if(aux != NULL){

info = aux->info;

Enqueue(&fila,info);//poe ele no final

}

return info;

}