#include <iostream>

using namespace std;

#include "LinkedList.h"

//-- Default constructor

LinkedList::LinkedList()

{

mySize = 0;

first = NULL;

}

//-- Definition of the copy constructor

LinkedList::LinkedList(const LinkedList & origList)

{

mySize = origList.mySize;

first = NULL;

if (mySize == 0)

return;

Node \* origPtr, \* lastPtr;

first = new Node(origList.first->data); // copy first node

lastPtr = first;

origPtr = origList.first->next;

while (origPtr != NULL)

{

lastPtr->next = new Node(origPtr->data);

origPtr = origPtr->next;

lastPtr = lastPtr->next;

}

}

//-- Definition of the destructor

LinkedList::~LinkedList()

{

Node \* prev = first;

Node \* ptr;

while (prev != NULL)

{

ptr = prev->next;

delete prev;

prev = ptr;

}

}

//-- Definition of insert()

void LinkedList::insert(ElementType dataVal, int index)

{

if (index < 0 || index > mySize)

{

cerr << "Illegal location to insert -- " << index << endl;

return;

}

mySize++;

Node \* newPtr = new Node(dataVal);

Node \* predPtr = first;

if (index == 0)

{

newPtr->next = first;

first = newPtr;

}

else

{

for(int i = 1; i < index; i++)

predPtr = predPtr->next;

newPtr->next = predPtr->next;

predPtr->next = newPtr;

}

}

//-- Definition of erase()

void LinkedList::erase(int index)

{

if (index < 0 || index >= mySize)

{

cerr << "Illegal location to delete -- " << index << endl;

return;

}

mySize--;

Node \* ptr;

Node \* predPtr = first;

if (index == 0)

{

ptr = first;

first = ptr->next;

delete ptr;

}

else

{

for(int i = 1; i < index; i++)

predPtr = predPtr->next;

ptr = predPtr->next;

predPtr->next = ptr->next;

delete ptr;

}

}

//-- Definition of display()

void LinkedList::display(ostream & out) const

{

Node \* ptr = first;

while (ptr != 0)

{

out << ptr->data << " ";

ptr = ptr->next;

}

}