#include <iostream>

#include <conio.h>

#include <assert.h>

using namespace std;

class nodeType

{

public:

int info;

nodeType\* link;

};

class linkedListType

{

public:

void initializeList()

{

destroyList();

}

bool isEmptyList()

{

if (count == 0)

return 1;

else

return -1;

}

int length()

{

return count;

}

void destroyList()

{

nodeType\* temp;

while (first != NULL)

{

temp = first;

first = first->link;

delete temp;

}

last = NULL;

count = 0;

}

int front()

{

assert(first != NULL); //assert is used to handle the out of range error.

return first->info;

}

int back()

{

assert(last != NULL);

return last->info;

}

bool search(const int& searchitem)

{

nodeType\* current;

bool found;

current = first; //current pointing to first node.

found = false;

while (current != NULL && !found)

{

if (current->info == searchitem)

found = true;

else

current = current->link;

}

return found;

}

void insertFirst(const int& newItem)

{

nodeType\* newNode;

newNode = new nodeType;

assert(newNode != NULL);

newNode->info = newItem;

newNode->link = first;

first = newNode;

count++;

if (last == NULL)

last = newNode;

//if list is empty new node will be first and as well as last node

}

void insertLast(const int& newItem)

{

nodeType\* newNode;

newNode = new nodeType;

assert(newNode != NULL);

newNode->info = newItem;

newNode->link = last;

last = newNode;

count++;

if (first == NULL)

first = newNode;

//if there are no nodes than last node will be like first node and as well as last node.

}

void deleteNode(const int& deleteItem)

{

nodeType\* current;

nodeType\* tailcurrent;

bool found;

if (first == NULL)

cout << "Can't delete an empty node" << endl;

else if (first->info == deleteItem)

{

current = first;

first = first->link;

count--;

if (first == NULL)

{

cout << "There was only one node in list";

last = NULL;

delete current;

}

}

else

{

//we will now have to traverse the list

found = false;

tailcurrent = first;

current = first->link;

while (current != NULL && !found)

{

if (current->info != deleteItem)

{

tailcurrent = current;

current = current->link;

}

else

found = true;

}

if (found)

{

tailcurrent->link = current->link;

count--;

if (last == current)

last = tailcurrent;

delete current;

}

else

cout << "The item to be deleted not found" << endl;

}

}

linkedListType()

{

first = NULL;

last = NULL;

count = 0;

}

~linkedListType()

{

destroyList(); //called destroy list function here which will properly delete list.

}

void print()const

{

nodeType\* current;

current = first;

while (current != NULL)

{

cout << current->info << endl;

current = current->link;

}

}

protected: int count;

nodeType\* first;

nodeType\* last;

};

void main()

{

linkedListType intList;

intList.initializeList();

if (intList.isEmptyList() == 1)

cout << "The list is empty " << endl;

cout << endl;

int counter;

int number;

cout << "Length of List: ";

cin >> counter;

cout << "Enter " << counter << " integers: ";

for (int indx = 0; indx < counter; indx++)

{

cin >> number;

intList.insertLast(number);

}

cout << endl;

cout << "The list you entered is: "; intList.print();

cout << endl << endl;

cout << "Enter number to be inserted at the front "; cin >> number;

intList.insertFirst(number);

cout << "The updated list is: "; intList.print();

cout << endl << endl; cout <<

"Length of List ";

cout << intList.length() << endl << endl;

cout << "Enter item to check its existence in list ";

cin >> number;

if (intList.search(number) == 1)

cout << "Item is present in the list " << endl;

else

cout << "Item not found in the list " << endl;

cout << endl;

cout << "Enter the item to be deleted : ";

cin >> number;

intList.deleteNode(number);

cout << "After removing number the list is: "; intList.print();

cout << endl << endl;

cout << "Displaying First item in the list "; cout <<

intList.front() << endl;

cout << "Displaying Last item in the list ";

cout << intList.back() << endl;

intList.destroyList(); intList.print();

\_getch();

}