/\*

\* List.cpp

\*

\* Timothy Lee

\* CIS 22C

\*

\*/

#include "List.h"

#include <iostream>

using namespace std;

List::List(): start(NULL), end(NULL), cursor(NULL), length(0) {}

List::~List()

{

cursor = start;

NodePtr temp;

while(cursor != NULL)

{

temp = cursor->next;

delete cursor;

cursor = temp;

}

}

//Make sure you understand: Why do I need the temp Node here?

//Because we need to know what node is the next before deleting current node.

void List::print()

{

NodePtr temp = start; //create a temporary iterator

while (temp != NULL) {

//What two lines of code go here?

cout << temp->data << " ";

temp = temp->next;

//Hint: One statement should contain a cout

}

cout << endl; //What does this do? move to next line after list is all printed.

}

void List::add\_start(int data)

{

if (length==0) //Why is this necessary? to check if the list is empty

{

start = new Node(data);

end = start;

}

else

{

NodePtr N = new Node(data);//create the new node by calling the node constructor

N->next = start;//set the new node's next field to point to the start

start = N;//make the start be the new node

}

length++;

}

void List::add\_end(int data)

{

if (length==0)

{

end = new Node(data);

start = end;

}

else

{

NodePtr N = new Node(data);

end->next = N;

end = N;

}

length++;

}

bool List::is\_empty()

{

return (length==0);

}

int List::get\_length()

{

return length;

}

int List::get\_start()

{

return start -> data;

}

int List::get\_end()

{

return end -> data;

}

/\*

\* List.h

\*

\* Timothy Lee

\* CIS 22C

\*

\*/

#ifndef LIST\_H\_

#define LIST\_H\_

#include <cstddef> //for NULL

class List

{

private:

struct Node

{

int data;

Node\* next;

Node(int data): next(NULL), data(data){}

};

typedef struct Node\* NodePtr;

NodePtr start;

NodePtr end;

NodePtr cursor;

int length;

public:

/\*\*Constructors and Destructors\*/

List();

//Default constructor; initializes and empty list

//Postcondition: List is empty set node to NULL

~List();

//Destructor. Frees memory allocated to the list

//Postcondition: nodes on the list is deleted and freed

List(const List &list);

//Copy construcor. Initializes list to have the same elements as another list

//Postcondition: New List is duplicated with old List list.

/\*\*Accessors\*/

int get\_start();

//Returns the first element in the list

//Precondition: There is a accessible int data in the list.

int get\_end();

//Returns the last element in the list

//Precondition: There is a accessible last data in the list.

int get\_cursor();

//Returns the element pointed to by the iterator

//Precondition: There is a accessible data that iterator is pointing

bool is\_empty();

//Determines whether a list is empty.

bool off\_end();

//Determines if the iterator is off the end of the list (i.e. whether cursor is NULL)

int get\_length();

//Returns the length of the list

/\*\*Manipulation Procedures\*/

void begin\_cursor();

//Moves the iterator to point to the first element in the list

//If the list is empty, the iterator remains NULL

//Postcondition: iterator will be pointed to fist element

void insert\_cursor(int data);

//Inserts a new element into the list in the position after the iterator

//Precondition: Iterator is pointed to certain element in the list

//Postcondition: Iterator will be pointed to new element in the list.

void remove\_end();

//Removes the value of the last element in the list

//Precondition: there is a element available in the list

//Postcondition: value of the last element in the list will be removed.

void remove\_start();

//Removes the value of the first element in the list

//Precondition: there is a first element in the list

//Postcondition: the first element in the list will be removed.

void add\_end(int data);

//Inserts a new element at the end of the list

//If the list is empty, the new element becomes both start and end

//Postcondition: a new element will be added to the end of the list

void add\_start(int data);

//Inserts a new element at the start of the list

//If the list is empty, the new element becomes both start and end

//Postcondition: a new element will be added at the start of the list.

void remove\_cursor();

//Removes the element pointed at by the iterator

//Precondition: iterator is pointing at some element

//Postcondition: iterator will be pointing at different element

void move\_cursor();

//Moves the iterator forward by one element in the list

//Precondition: iterator is pointed at the element

//Postcondition: iterator will be pointed at the other element

/\*\*Additional List Operations\*/

void print();

//Prints to the console the value of each element in the list sequentially

//and separated by a blank space

//Prints nothing if the list is empty

};

#endif /\* LIST\_H\_ \*/

/\*

\* ListTest.cpp

\*

\* Timothy Lee

\* CIS 22C

\*

\*/

#include <iostream>

#include "List.h"

using namespace std;

int main()

{

//creating a new list object L

List L;

L.add\_start(5);

cout << "The start of the List should be 5: " << L.get\_start() << endl;

cout << "The end of the List should be 5: " << L.get\_end() << endl;

cout << "List should contain: 5" << endl;

L.print();

L.add\_end(10);

cout << "The start of the List should be 5: " << L.get\_start() << endl;

cout << "The end of the List should be 10: " << L.get\_end() << endl;

cout << "List should contain: 5 10" << endl;

L.print();

L.add\_start(12);

cout << "The head of the List should be " << L.get\_start() << endl;

cout << "The tail of the List should be " << L.get\_end() << endl;

cout << "List should contain: 5, 10, 12" << endl;

L.print();

L.add\_end(222);

cout << "The head of the List should be " << L.get\_start() << endl;

cout << "The tail of the List should be " << L.get\_end() << endl;

cout << "List should contain: 5, 10, 12, 222" << endl;

L.print();

cout << "The length of the List should be 4." << endl;

cout << "The length of the List: " << L.get\_length() << endl;

cout << "The list should not be empty." << endl;

cout << "Is the list empty? ";

if(L.is\_empty() == 0)

{

cout << "no" << endl;

}

}

/\*

The start of the List should be 5: 5

The end of the List should be 5: 5

List should contain: 5

5

The start of the List should be 5: 5

The end of the List should be 10: 10

List should contain: 5 10

5 10

The head of the List should be 12

The tail of the List should be 10

List should contain: 5, 10, 12

12 5 10

The head of the List should be 12

The tail of the List should be 222

List should contain: 5, 10, 12, 222

12 5 10 222

The length of the List should be 4.

The length of the List: 4

The list should not be empty.

Is the list empty? no