List.h

#ifndef LIST\_H

#define LIST\_H

#include <string>

using namespace std;

class List;

class Tester;

class Node

{

public:

/\*\*

Constructs a node with a given data value.

@param element the data to store in this node

\*/

Node(string element);

private:

string data;

Node\* previous;

Node\* next;

friend class List;

friend class Iterator;

friend class Tester;

};

class Iterator

{

public:

bool past\_end() const;

void next();

string get() const;

private:

Node\* position;

List\* container;

friend class List;

friend class Tester;

};

class List

{

public:

/\*\*

Constructs an empty list.

\*/

List();

/\*\*

\*

Adds an element to the back of the list.

@param element the value to add

\*/

void push\_back(string element);

Iterator begin();

void insert(Iterator iter, string element);

private:

Node\* first;

Node\* last;

friend class Tester;

};

#endif

Tester.cpp

#include "List.h"

#include <iostream>

#include <iomanip>

using namespace std;

class Tester

{

public:

void test1()

{

List lst;

lst.push\_back("Fred");

lst.push\_back("Wilma");

lst.push\_back("Barney");

Iterator iter = lst.begin();

iter.next();

lst.insert(iter, "Betty");

cout << lst.first->data << endl;

cout << "Expected: Fred" << endl;

cout << lst.first->next->data << endl;

cout << "Expected: Betty" << endl;

cout << lst.first->next->next->data << endl;

cout << "Expected: Wilma" << endl;

cout << iter.position->data << endl;

cout << "Expected: Wilma" << endl;

cout << iter.position->previous->data << endl;

cout << "Expected: Betty" << endl;

cout << iter.position->previous->previous->data << endl;

cout << "Expected: Fred" << endl;

}

void test2()

{

List lst;

lst.push\_back("Fred");

lst.push\_back("Wilma");

lst.push\_back("Barney");

Iterator iter = lst.begin();

lst.insert(iter, "Betty");

cout << lst.first->data << endl;

cout << "Expected: Betty" << endl;

cout << lst.first->next->data << endl;

cout << "Expected: Fred" << endl;

cout << lst.first->next->next->data << endl;

cout << "Expected: Wilma" << endl;

cout << iter.position->data << endl;

cout << "Expected: Fred" << endl;

cout << iter.position->previous->data << endl;

cout << "Expected: Betty" << endl;

cout << iter.position->previous->previous << endl;

cout << "Expected: 0" << endl;

}

};

int main()

{

Tester tester;

tester.test1();

tester.test2();

return 0;

}

#include <string>

#include "List.h"

using namespace std;

Node::Node(string element)

{

data = element;

previous = nullptr;

next = nullptr;

}

List::List()

{

first = nullptr;

last = nullptr;

}

Iterator List::begin()

{

Iterator iter;

iter.container = this;

iter.position = first; // Points to first element

return iter;

}

void List::insert(Iterator iter, string element)

{

if (iter.position == nullptr)

{

push\_back(element);

return;

}

Node\* after = iter.position;

Node\* before = after->previous;

Node\* new\_node = new Node(element);

after->previous=new\_node;

if(before == nullptr)

{

new\_node->previous = nullptr;

first = new\_node;

}

else{

new\_node ->previous = before;

before ->next = new\_node;

}

new\_node -> next = after;

}

// We aren't showing you the implementation of List::push\_back

// and the Iterator member functions