/// Andrew Souza

/// Comp 210 -- Spring 2024

/// Assignment 2 - Problem 1

#include <iostream>

#include "list.h"

using namespace std;

/\*

For this assignment I’ve opted to copy the demo classes into 2 separate folders.

For 14.11.5 the demo class is used as-is, and thus I’ll exclude it from the submission.

For 14.12.1 I will include the class functions I’ve modified. In most cases I’ve simply

substituted the Node\* last for first->previous.

\*/

/\*

Programming Project 14.12.1: ••.

Turn the linked list implementation into a circular list:

Have the previous pointer of the first node point to the last node,

and the next pointer of the last node point to the first node.

Then remove the last pointer in the List class because the value

can now be obtained as first->previous.

Reimplement the member functions so that they have the same effect as before.

\*/

List::List()

{

first = nullptr;

}

void List::push\_back(string element)

{

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

if (first == nullptr) // List is empty

{

first = new\_node;

first->next = first; // First initializes itself

first->previous = first; // as both next and previous

}

else

{

Node\* temp = first->previous;

temp->next = new\_node;

new\_node->previous = temp;

new\_node->next = first;

first->previous = new\_node;

}

}

Iterator List::end()

{

Iterator iter;

iter.position = first->previous;

iter.container = this;

return iter;

}

void Iterator::previous()

{

position = position->previous;

}

int main() {

List names;

string name = "n/a";

int namesCount = 0;

cout << "Input a name. type -1 to quit" << endl;

cin >> name;

while (name != "-1") {

names.push\_back(name);

namesCount++;

cin >> name;

}

Iterator pos = names.begin();

for (int i = 0; i < namesCount; i++) {

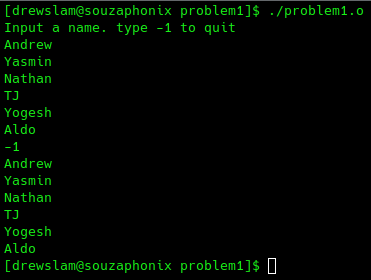
cout << pos.get() << endl;

pos.next();

}

return 0;

}



/// Andrew Souza

/// Comp 210 -- Spring 2024

/// Assignment 2 - Problem 2

#include <iostream>

#include <string>

#include "list.h"

using namespace std;

/\*

Practice Exercise 14.11.5: ••.

Write a function merge that merges two lists into one,

alternating elements from each list until the end of

one of the lists has been reached, then appending the

remaining elements of the other list.

For example, merging the lists containing A B C and

D E F G H should yield the list A D B E C F G H.

\*/

List merge(List a, List b, int count) {

Iterator pos1 = a.begin(); // 2 Iterators are initialized for each list

Iterator pos2 = b.begin();

List result; // A result list is defined to hold the value of both lists

int i = 0;

do {

if (!pos1.equals(a.end())) { // As long as each Iterator holds a value then the value is pushed

result.push\_back(pos1.get()); // into the result list

pos1.next();

}

if (!pos2.equals(b.end())) {

result.push\_back(pos2.get());

pos2.next();

}

i++;

} while (i != count); // i increases until it equals the count value

return result;

}

int main() {

List list1;

List list2;

int count = 0;

list1.push\_back("A");

list1.push\_back("B");

list1.push\_back("C");

list2.push\_back("D");

list2.push\_back("E");

list2.push\_back("F");

list2.push\_back("G");

list2.push\_back("H");

// Each linked list is looped through to increase the value of count

// until we reach the size of our result list

for (Iterator pos = list1.begin(); !pos.equals(list1.end()); pos.next()) {

count++;

}

for (Iterator pos = list2.begin(); !pos.equals(list2.end()); pos.next()) {

count++;

}

List list3 = merge(list1, list2, count);

for (Iterator pos = list3.begin(); !pos.equals(list3.end()); pos.next()) {

cout << pos.get() << " ";

}

cout << endl;

return 0;

}

