1. Download the following program code:

<https://markbowman.org/231/Lab16.zip>

The project implements a circular linked list. The last node in the list is linked to the first. The list class has two pointers, head and current. The head pointer stays put, while the current pointer can move around the list.

Diagram

Description automatically generated

You may not change the List.h or Node.h files, or any of the existing class functions.

2. Using the loop in the list destructor as an example, implement the list::display() function. It should start at head and loop until it reaches the first node again.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Display Function for List Class

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//display the node

void list::display(ostream& out)

{

node\* p;

if (head != NULL) // non-empty list

{

p = head; // set p to head

// loop till we reach the last node of the list

while (p->next != head)

{

p->put(out); // output node p using put function of node class

p = p->next; // move to the next node

}

p->put(out); // output last node

current->put(out); // output current node

}

else // empty list

out << "Empty list" << endl;

}

3. Test your code by removing the /\* Step 3 \*/ comments in Lab16.cpp. Compile and run the program. If your display() function is correct, you should see the output below:

Enter file name: ***Lab16.txt***

00162980 One 00162D40

00162D40 Two 00162D90

00162D90 Three 00162C50

00162C50 Four 00162B60

00162B60 Five 00162DE0

00162DE0 Six 001629D0

The last next is the address of the first node

001629D0 Seven 00162C00

00162C00 Eight 00163100

00163100 Nine 00162E30

00162E30 Ten 00162980

4. The insert() function uses the current pointer to add nodes at the end of the list. Update the display() function to show the node at the current position.

5. Compile and run the program again. You should see the current node, which is the last one added to the list. Save your output.

Enter file name: Lab16.txt

00171470 One 001710B0

001710B0 Two 00170FC0

00170FC0 Three 001714C0

001714C0 Four 00171600

00171600 Five 00171510

00171510 Six 00171010

00171010 Seven 00170F70

00170F70 Eight 00171560

00171560 Nine 001711F0

001711F0 Ten 00171470

001711F0 Ten 00171470

Text

Description automatically generated

6. Implement the list::rotate() function. It should move the current pointer n nodes around the list.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Rotate Function for List Class

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//rotate the current pointer n nodes around the list

void list::rotate(int val)

{

int i; //counter

node\* p = head;

for (i = 0; i < val - 1; i++) //user specified value for n nodes

{

p = p->next; //rotate from current pointer to next pointer

}

current = p; //current pointer after rotations

}

7. Test your code by removing the /\* Step 7 \*/ comments in Lab16.cpp. Compile and run the program enough times to verify that it works.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

8. Implement the list::forward() function with the following code:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Forward

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void list::forward(ostream &out)

{ out << "Forwards" << endl;

out << "--------" << endl;

// Check list

if(head!=NULL) current->forward(out,current);

}

This is a stub function, that checks that the list has nodes in it, then calls a node function at the current position to do the actual work.

9. Implement the node::forwards() function. It should display the node, then recurse if its next pointer isn’t the same as the argument pointer.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Forward Function for Node Class

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void node::forward(ostream& out, node\* current)

{

//display the node, recurse if next pointer

//is not same as argument pointer

put(out);

if (next != NULL && next->name != current->name)

{

//print node forward from current pointer

next->forward(out, current);

}

}

10. Test your code by removing the /\* Step 10 \*/ comments in Lab16.cpp. Compile and run the program with different rotation values to verify that it works. Save your output from three runs.

Sample Run

Enter file name: ***Lab16.txt***

Enter rotations: 14

001B2918 One 001B2968

001B2968 Two 001B26E8

001B26E8 Three 001B2738

001B2738 Four 001B2B98

001B2B98 Five 001B2DC8

001B2DC8 Six 001B2A08

001B2A08 Seven 001B2CD8

001B2CD8 Eight 001B2788

001B2788 Nine 001B27D8

001B27D8 Ten 001B2918

001B2738 Four 001B2B98

Forwards

--------

001B2738 Four 001B2B98

001B2B98 Five 001B2DC8

001B2DC8 Six 001B2A08

001B2A08 Seven 001B2CD8

001B2CD8 Eight 001B2788

001B2788 Nine 001B27D8

001B27D8 Ten 001B2918

001B2918 One 001B2968

001B2968 Two 001B26E8

001B26E8 Three 001B2738

Text

Description automatically generated

Extra Credit

Implement the list::backward() and node::backward() functions. Print a copy of both functions, and three sample runs.

**List:Backward() Function:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Backward Function for List Class

\* Extra Credit

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void list::backward(ostream& out)

{

//print a list backwards

out << "Backwards" << endl;

out << "--------" << endl;

//check list to see if at the end of list

if (head != NULL)

{

//current to backward

current->backward(out, current);

}

}

**Node:Backward() Function:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Extra Credit

\* Backward Function for Node Class

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void node::backward(ostream& out, node\* current)

{

//print the list backward of each node, recurse

//if the next pointer is not same as argument

//pointer

if (next != NULL && next->name != current->name)

{

next->backward(out, current);

}

//display the node (backwards)

put(out);

}

**Sample Run 1:**

**Text

Description automatically generated**

**Sample Run 2:**

**Text

Description automatically generated**

**Sample Run 3:**

**Text

Description automatically generated**

**Codes:**

**Node.h:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Node.h

\* Written by Mark M Bowman

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class list;

class node

{ friend class list;

public:

node(string arg=""); // Constructor

void put(ostream& out); // Output node

void forward(ostream &,node \*); // Recursive forwards output

void backward(ostream &,node \*); // Recursive backwards output

private:

string name; // Data

node\* next; // Link to the next node

};

**Node.cpp:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* List.cpp

\* Created by Twymun Safford

\* CPSC 231-MW

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

#include "List.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Constructor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

list::list()

{ head = NULL;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Destructor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

list::~list()

{ node \*p,\*q;

// Check list

if(head==NULL) return;

// Loop through list

p = head;

do { q = p;

p = p->next;

delete q;

} while(p!=head);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Insert

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool list::insert(string arg)

{

node \*p;

// Create a node

p = new node(arg);

if(p==NULL) return false;

// Add to empty list

if(head==NULL)

{ head = current = p;

p->next = head;

return true;

};

// Add at current position

p->next = current->next;

current->next = p;

current = p;

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Forward

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void list::forward(ostream& out)

{

out << "Forwards" << endl;

out << "--------" << endl;

// Check list

if (head != NULL)

{

current->forward(out, current);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Backward Function for List Class

\* Extra Credit

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void list::backward(ostream& out)

{

//print a list backwards

out << "Backwards" << endl;

out << "--------" << endl;

//check list to see if at the end of list

if (head != NULL)

{

//current to backward

current->backward(out, current);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Rotate Function for List Class

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//rotate the current pointer n nodes around the list

void list::rotate(int val)

{

int i; //counter

node\* p = head;

for (i = 0; i < val - 1; i++) //user specified value for n nodes

{

p = p->next; //rotate from current pointer to next pointer

}

current = p; //current pointer after rotations

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Display Function for List Class

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//display the node

void list::display(ostream& out)

{

node\* p;

if (head != NULL) // non-empty list

{

p = head; // set p to head

// loop till we reach the last node of the list

while (p->next != head)

{

p->put(out); // output node p using put function of node class

p = p->next; // move to the next node

}

p->put(out); // output last node

cout << "\n";

current->put(out); // output current node

}

else // empty list

out << "Empty list" << endl;

}

**List.h:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Node.h

\* Written by Mark M Bowman

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "Node.h"

class list

{ public:

list(); // Constructor

~list(); // Destructor

bool insert(string args); // Add a node to list

void rotate(int); // Rotate the list

void display(ostream &); // Display entire list

void forward(ostream &); // Display forwards list

void backward(ostream &); // Display backwards list

private:

node\* head; // Link to the head node

node\* current; // Link to the current node

};

**List.cpp:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* List.cpp

\* Created by Twymun Safford

\* CPSC 231-MW

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

#include "List.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Constructor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

list::list()

{ head = NULL;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Destructor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

list::~list()

{ node \*p,\*q;

// Check list

if(head==NULL) return;

// Loop through list

p = head;

do { q = p;

p = p->next;

delete q;

} while(p!=head);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Insert

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool list::insert(string arg)

{

node \*p;

// Create a node

p = new node(arg);

if(p==NULL) return false;

// Add to empty list

if(head==NULL)

{ head = current = p;

p->next = head;

return true;

};

// Add at current position

p->next = current->next;

current->next = p;

current = p;

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Forward

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void list::forward(ostream& out)

{

out << "Forwards" << endl;

out << "--------" << endl;

// Check list

if (head != NULL)

{

current->forward(out, current);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Backward Function for List Class

\* Extra Credit

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void list::backward(ostream& out)

{

//print a list backwards

out << "Backwards" << endl;

out << "--------" << endl;

//check list to see if at the end of list

if (head != NULL)

{

//current to backward

current->backward(out, current);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Rotate Function for List Class

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//rotate the current pointer n nodes around the list

void list::rotate(int val)

{

int i; //counter

node\* p = head;

for (i = 0; i < val - 1; i++) //user specified value for n nodes

{

p = p->next; //rotate from current pointer to next pointer

}

current = p; //current pointer after rotations

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Display Function for List Class

\* Twymun Safford

\* CPSC 231

\* Date:10/21/2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//display the node

void list::display(ostream& out)

{

node\* p;

if (head != NULL) // non-empty list

{

p = head; // set p to head

// loop till we reach the last node of the list

while (p->next != head)

{

p->put(out); // output node p using put function of node class

p = p->next; // move to the next node

}

p->put(out); // output last node

cout << "\n";

current->put(out); // output current node

}

else // empty list

out << "Empty list" << endl;

}

**Lab16.cpp:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 16

\* Created by Mark M Bowman

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

#include "List.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{ int n;

string name;

fstream infile;

list a;

// Open file

cout << "Enter file name: ";

cin >> name;

cout << left << endl;

infile.open(name,ios::in);

// Loop through file

while(!infile.eof())

{ infile >> name;

// Process valid input

if(infile.good())

a.insert(name);

};

// Close file

infile.close();

//Step 7

cout << "Enter rotations: ";

cin >> n;

cout << endl;

a.rotate(n);

//Step 3

a.display(cout);

cout << endl;

//Step 10

a.forward(cout);

cout << endl;

//Extra credit - backwards

a.backward(cout);

cout << endl;

}