CSCE 221 Cover Page

Programming Assignment #3

Kirsten Madina 626003641

Kirsten.madina Kirsten.madina@tamu.edu

Please list all sour es in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of Sources		
People	Piazza forum	
Web pages	Stack overflow: https://stackoverflow.com/questions/3928935/size-of-linked-list-in-c-function	Cplusplus: http://www.cplusplus.com/doc/tutorial/exceptions/
Printed material	None	
Other Sources	None	

I certify that I have listed all the sources that I used to develop the olutions/code to the submitted work.

"On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."

Kirsten Madina March 7th 2019

1. Description of an assignment problem.

The problem consists of 3 parts. The first part is to implement a DoublyLinkedList class with node objects of type 'int'. The second part is to template the DoublyLinkedList class in order to take node objects of any type. The last part is to use the adapter method to created a MinQueue class using the templated DoublyLinkedList as a member variable of the class, able to store comparable elements of type T such as integers, characters, or floats.

2. The description of data structures…

a. Data structures using ADT's

There are three data structures here: A Doubly Linked List Node, A Doubly Linked List, and a Queue.

The Doubly Linked List Node has 3 private member variables: an element containing data, a pointer to the previous node, and a pointer to the next node. It has no functions that can act on it. The next data structure is a Doubly Linked List.

It's default state contains two Node (referred to as a 'header' and 'trailer' in this implementation), and Nodes which are added to the Linked list are added somewhere after the header and before the trailer. In this implementation, the DLL has functions that allow you to insert nodes into the list (to the first position, to the last position, and to somewhere in the middle as long as there's a reference pointer), to remove nodes (with the same three conditions), getter functions that access the first node, the trailer node, and the first&last objects, a Boolean to return if the DLL is empty, copy constructors, and move constructors.

This Queue implementation data structure has a member variable Doubly Linked list and an integer member referring to the minimum value in the Queue. It has functions to enqueue(), dequeue(), and min() to return the value of the minimum.

b. Algorithms to solve the problem

insertFirst()/insertAfter() for DLL have the same principle idea. InsertFirst() inserts a node after the header whilst insertAfter() inserts a node after some reference node p. For simplicity, we are going to refer to the header as p in this description. To implement this function, I created a pointer to a Node using the 'new' keyword and constructed the node using a new data member (newobj), where Node.prev is p and Node.next is p.next. We then reassign p.next->prev to point to the new Node and p.next to point to the new Node.

Similarly, **insertLast()/insertBefore**() for DLL have the same idea, where InsertLast() inserts a node after the trailer and insertBefore() inserts a node before some reference node p. (Trailer will be referred to as p). To implement this function, we create a Node using 'new' and constructed the node with a data member (newobj), where Node.prev is p.prev and Node.next is p. We then reassign p.prev->next to the Node and p.prev to the Node.

removeFirst()/removeAfter() are implemented by assigning a DListNode pointer ('after') to p.next (or header.next) and reassigning p.next to after->next and reassigning the node after the node we want to remove (after->next->prev) to point to p. We then return the object of the node we just removed and delete 'after' to avoid a leak.

removeLast()/removeBefore() are implemented similarly, however we assign the Node 'before' to p.prev and reassign p.prev to point to before->prev and before->prev->next to p. It also returns the objext of the node that was removed and deletes 'before'.

A function **DoublyLinkedListLength(DoublyLinkedList& dll)** sets a DListNode pointer 'current' to the first Node of dll. It uses a while loop with the condition that the current node is not the trailer node to iterate through the Linked List. As it iterates, an integer referring to count post increments by 1. The function then returns the count.

The ostream operator<< iterates through the linked list the same way, however is outs the current Node's object into the ostream& object 'out' and returns out.

To **dequeue()** in MinQueue, we iterate through the LinkedList member and compare each of the Node's objects linearly in order to find the minimum value, and the DLL removeAfter() function is called in order to remove the min value node.

The MinQueue **min()** function outputs the minimum value object of all the node's by iterating through the linked list and comparing all the objects linearly.

c. Algorithm Analysis

Since all algorithms save for dequeue(), min(), operator<<, and DoublyLinkedListLength(), simply calls, assignments, and reassignments, they are all O(1). The four algorithms above iterate through the linked list using a single while loop or four loop and increment by 1 and therefore are linear or O(n).

3. C++ Organization and Implementation

a. Classes and interfaces

Classes: DListNode, DListNode<T> DoublyLinkedList, DoublyLinkedList<T>, MinQueue.

Interfaces: getter functions such as MinQueue::getdll() DLL::*getFirst(), getAfterLast(), first(), last(), and all the public functions listed above.

b. Inheritance/Polymorphism

1) A templated version of the integer implementation of a Doubly Linked List was created in Part1. In Part2, Minqueue had DoublyLinkedList<T> as a data member in accordance to the adapter method.

4. User Guide

a. Compilation and Execution:

>cd madina_kirsten_PA3\DoublyLinkedList

>make all

>./run-dll

>cd madina_kirsten_PA3\TemplatedDoublyLinkedList

>make

>./run-tdll

>cd madina_kirsten_PA3\MinQueue

> g++ MinQueue.cpp -std=c++11

>./a.out

5. Input/Output

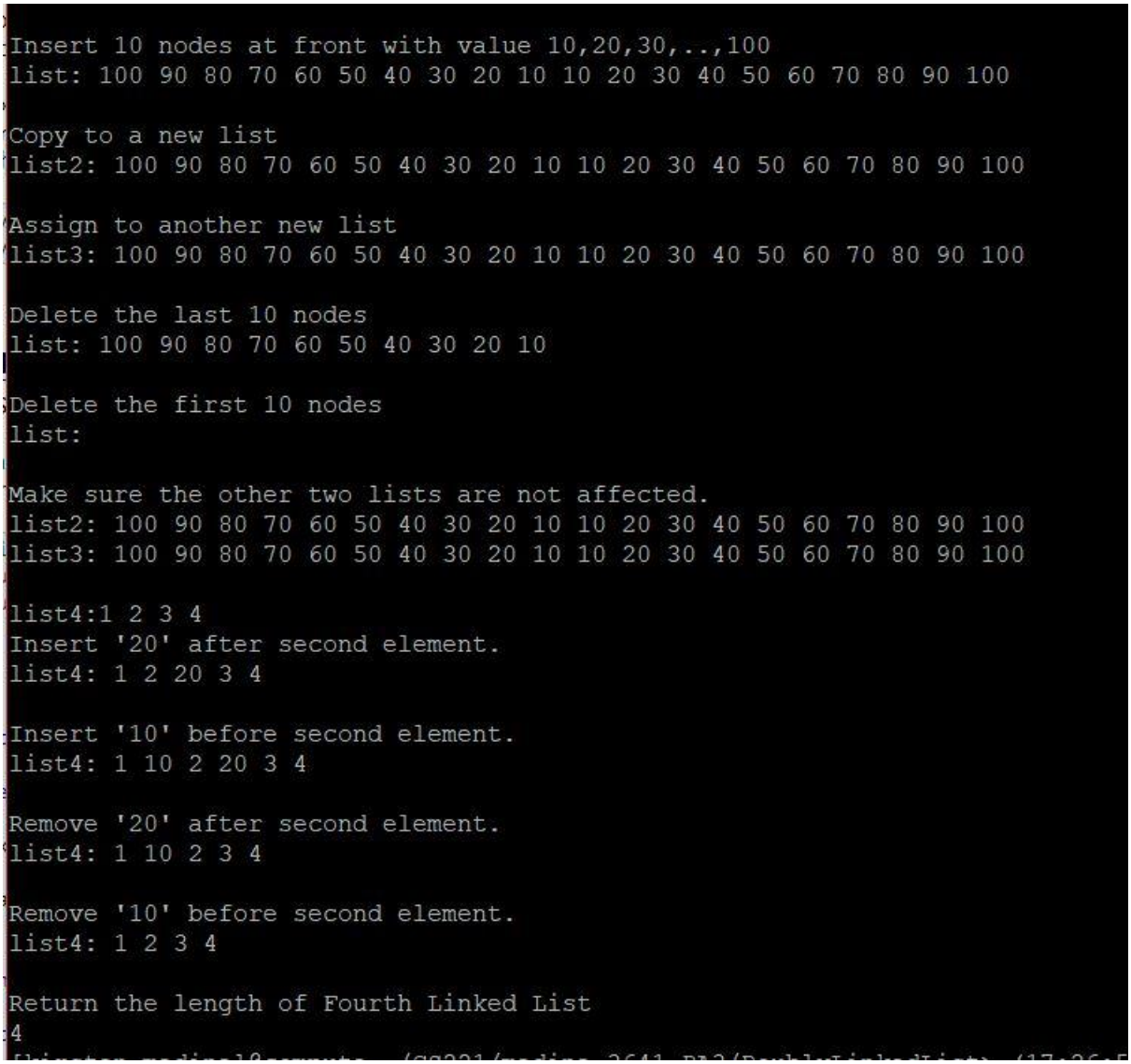
No input or output files are included with this programming assignment.

6. Exceptions

Exceptions were used in order to detect cases of 1) trying to remove nodes from empty linked lists, 2) trying to access the first or last object of an empty list.

7. Testing

Proof of testing DoublyLinkedList:



Proof of testing Templated:

```
g++-8.2.0 -std=c++11 TemplateMain.cpp -o run-tdll

[kirsten.madina]@compute ~/CS221/madina_3641_PA3/TemplateDoublyLinkedList> (1
0:29 03/07/19)
:: ./run-tdll
Create a new list
list:

Insert 10 nodes at back with value 10,20,30,..,100
list: 10 20 30 40 50 60 70 80 90 100

Insert 10 nodes at front with value 10,20,30,..,100
list: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Copy to a new list
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Assign to another new list
list3: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

Delete the last 10 nodes
list: 100 90 80 70 60 50 40 30 20 10

Delete the first 10 nodes
list:

Make sure the other two lists are not affected.
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100
list3: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

list4:1 2 3 4
Insert '20' after second element.
list4: 1 2 20 3 4

Insert '10' before second element.
list4: 1 10 2 20 3 4

Remove '20' after second element.
list4: 1 10 2 3 4

Remove '10' before second element.
list4: 1 2 3 4

Return the length of Fourth Linked List
4
```

Proof of Testing MinQueue:

```
Create a new list of integers:
List:

Enqueue 10 nodes with random values:
List: 3 6 17 15 13 15 6 12 9 1
Size of List: 10

Dequeue 3 minimum values:
min: 1
List: 3 6 17 15 13 15 6 12 9
min: 3
List: 6 17 15 13 15 6 12 9
min: 6
List: 17 15 13 15 6 12 9
Size of List: 7

Create a new list of characters:
List:

Enqueue 10 random letters:
List:
c d a r z o w k k y

Dequeue 2 minimum values:
min: a
List: c d r z o w k k y
min: c
List: d r z o w k k y
Size of List: 8
```