Programming Assignment 3 (part 1)

Syed Abeer Hasan Zaidi

March 7, 2019

Program Description and Purpose of Assignment

The assignment was to implement a doubly linked list that stores integers, and then implement a templated variant of that doubly linked list which should be able to handle any data type. The purpose of this assignment was to teach us about doubly linked lists and to reinforce what we had learned about them during the lectures.

The Data Structure

The Data Structures are an unsorted doubly linked list and an unsorted templated doubly linked list. Both linked lists have headers and trailers on the top and bottom of them, in order to make traversing the list, and moving the list, quicker than normal. Also the header and trailer gives the doubly linked list a more definitive structure than it would have without it.

Compiling and Running Instructions

I compiled this program using the provided makefiles, however the makefile for the templated doubly linked list wasn't working so I had to make a new one based off the one for the original doubly linked list. The call to run for the doubly linked list is ./run-dll, and ./run-tdll for the templated version. When running, the program performs a series of tests to verify that every part of the class is working as expected. This includes, copy construction, copy assignment, deleting elements, inserting new elements, and clearing the linked list entirely.

Logical Exceptions and Bugs

The templated doubly linked list has one bug where it will only take in the desired input after its DListNode and constructor is edited where the empty object is indicative of the desired input. The program will compile, but when it runs I encountered a segmentation fault, which goes away when we change e

= 0, header(0) and trailer(0) to e = “”, header(“”), trailer (“”). This bug seems to be restricted to when we make a queue of strings.

Linked List Implementation

The linked list was implemented with heavy utilization of a struct called DListNode. This allowed me to traverse the doubly linked list in both directions. When the list was initially constructed, it was empty, and it was later populated using the new call. Each node in the list was allocated on the heap. At the end of the programs runtime the list would be cleared by looping through the list and deleting each node. The insertAfter, insertBefore, removeAfter, and removeBefore fucntions worked in a similar way. They looped through the list while comparing the value of obj in a specific DListNode with every node in the list. If the remove functions were unable to find a similar Node, they would return a 0 instead of the value of obj in the removed node.

Complexity Analysis

Almost every function in both Doubly Linked Lists had an average time complexity of O(n), where n is the number of elements in the list. All those functions used a while loop in order to travese the list. The remaining funcitons, the default constructor, move constructor, move assignment operator, and the getters in the class, all have a time complexity of O(1) as their runtime isn't affected by the number of elements in the list.

Testing Evidence

Figure 1: Original Doubly Linked List Testing

Figure 2: Templated Doubly Linked List Testing

```
/Users/abeerzaidi/Desktop/CS221/PA3
Abeers-MacBook-Pro-4:PA3 abeerzaidi$ cd DoublyLinkedList
Abeers-MacBook-Pro-4:DoublyLinkedList abeerzaidi$ ./run-dll
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


List 1 Length: 10
Delete the first 10 nodes
list:


List 1 Length: 0
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

List 2, Insert 19 after 20:
100 90 80 70 60 50 40 30 20 19 10 10 20 30 40 50 60 70 80 90 100

List 3, Insert 19 before 20:
100 90 80 70 60 50 40 30 19 20 10 10 20 30 40 50 60 70 80 90 100

List 2 Length: 21
List 3 Length: 21

List 2, Remove 19 after 20:
100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

List3, Remove 19 before 20:
100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

List 2 Length: 20
List 3 Length: 20

Delete Lists 2 and 3
List 2 Length: 0
List 3 Length: 0

Abeers-MacBook-Pro-4:DoublyLinkedList abeerzaidi$
```

Figure 1:

```
Abeers-MacBook-Pro-4:DoublyLinkedList abeerzaidi$ cd -
/Users/abeerzaidi/Desktop/CS221/PA3
Abeers-MacBook-Pro-4:PA3 abeerzaidi$ cd TemplateDoublyLinkedList
Abeers-MacBook-Pro-4:TemplateDoublyLinkedList abeerzaidi$ ./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

List 2, Insert 19 after 20:
100 90 80 70 60 50 40 30 20 19 10 10 20 30 40 50 60 70 80 90 100

List 3, Insert 19 before 20:
100 90 80 70 60 50 40 30 19 20 10 10 20 30 40 50 60 70 80 90 100

List 2 Length: 21
List 3 Length: 21

List 2, Remove 19 after 20:
100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

List3, Remove 19 before 20:
100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100

List 2 Length: 20
List 3 Length: 20

Delete Lists 2 and 3
List 2 Length: 0
List 3 Length: 0

Abeers-MacBook-Pro-4:TemplateDoublyLinkedList abeerzaidi$
```

Figure 2: