Programming Assignment 3

Josh Zschieshce

CSCE 221-505

UIN:523000614

March,6,2016

Part1: Program Description

1.1)

We manipulated doubly linked lists of intigers, and needing to test and implement the functions that we were supposed to implement.

1.2)

Users should be able to search through a list of books by title, input new books into the list if no book is found, and out all of the books to the screen.

Part2: Purpose of the Assignment

The purpose of this assignment is to make us understand the concepts of Doubly Linked Lists, and to implement those concepts in the form of a program.

Part3: Data Structure Description

3.1)

Linked Lists and Doubly Linked lists allow the programmer(s) to hold a lot of data about various aspects of an object or many objects themselves. One of the most interesting features is that Lists can be spread through the memory, unlike vectors or arrays. Lists are much more efficient, as only a few addresses have to be removed/rewritten where a large vector would have to move many elements.

3.2)

The implementation of the Part 1 was not that complicated, except for the Templated DoublyLinkedList. And I could not successfully implement the Part 2.

Part 4: The implementation of Doubly Linked Lists

4.1)

A) The first assignment was to get the functions insert_before, insert_after, delete_before, delete_after implemented within the provided SimpleDoublyLinkedList source file.

   I) Insert_before runs in O(1) time and a node is added at the index of the node before the index specified. The pointers to the old nodes must be rewritten to now point to the inserted node so that the list is maintained.

   II) Insert_after runs in O(1) time and a node is added at the index of the node after the node specified. Again, the pointers to the old nodes must be rewritten so that the list is maintained.

   III) Delete_before runs in O(1) time and a node is removed from the list at the index before the index specified. Again, the pointers to the old nodes must be rewritten so that the list is maintained. This had one condition being that elements cannot be removed at the beginning of the list.

   IV) Dekete_after runs in O(1) time and a node is removed from the list at the index after the index specified. Again, the pointers to the old nodes must be rewritten so that the list is maintained. This also had the same condition as Delete_before.

   V) After Analyzing, I see that the code runs and that there appear to be no errors.

B) The next part of the assignment was to get the non-templated DoublyLinkedList source code working correctly by implementing the copy constructor, overloading the operator = and overloading the operator <<.

   I) In the Copy Constructor, which runs in O(n) time, A new variable was initialized of type list of integers. Data then in passed to the constructor to create a new variable with meaningful data.

II) In the Assignment operator =, which runs in O(n) time, is in essence the same as the normal, un-overloaded assignment operator. How this works is that it deletes the entire list, then inserts the first element, then traverses the list simply inserting the elements using the insertLast() function to insert after each node

III) In the Output operator <<, which runs in O(n) time, is in essence the same as the normal, un-overloaded output operator. It works by first checking if the list is empty, if so simply out that the list is empty. Then makes a new DListNode initialized to the element in the first index, then traverses the list outing to the screen as it goes.

IV) After Analyzing, I see that the code runs and that there appear to be no errors.

C) The next part of the assignment was to template the DoublyLinkedList .cpp source file. This was templated only after the first DoublyLinkedList.cpp source file was suffessfully implemented. To template the DoublyLinkedList.cpp, it was simply copied over and the lines template<typename T> was added before every function and class and every function and variable had to be typenamed T. This was by far the code that took the longest to implement. One of the most significant gurdles was realizing that T obj was not initialized, th solution was to use the syntax T obj = T();. After analyzing, the code works as predicted with no errors

4.2)

Due to many tests across multiple classes, and time constraints, and the time it took to debug the TemplatedDoublyLinkedList, I was unable to implement the library database. I was able to get the library database to compile, however upon trying to execute the program, it simply has a segmentation fault. If the segmentation fault issue could've been debugged, then I feel like the program would immediately execute correctly. I also feel like the segmentation fault occurs within the insertOrderly function. Theoretically, the insertOrderly() function runs in O(n) time. Theoretically, the search_through() functions searches through the entire list for a specific title, author, and isbn, keeps track of a count variable c and iterates that if the variables mentioned previously match something in the list, otherwise it will tell you that there were no elements matching the search in the list, this runs in O(n) time. The function display_books simply goes through the vector of lists, checks to see what parts of the list is false, if so,

continue on, else use the overloaded operator to out the elements in the vector of lists, it has runtime of O(n). The function search_for_book is where the user actually enters in what they would like to search for e.g. author, title, isbn, etc. the function runs in O(1) time. I am unsure of how many hours I spent on this code , but it was many. The definitions of the Record classs are in the Record.cpp file, all member functions purpose is apparent, and all functions run in O(1) time.

Part 5: Program Orginizaton/Classes

5.1) Part 1:

Simple Doubly Linked Lists: All code is located in one file and contains the main, all functions, and all declarations and definitions.

Un-templated Doubly Linked Lists: the main was provided, and contains all of the test cases that Dr. Leyk would like us to use. DoublyLinkedLists.h contains all of the declarations for the functions used in the DoublyLinkedLists.cpp file that contains all of the corresponding definitions.

Templated Doubly Linked Lists: TemplateMain.cpp was provided, and contains all of the test cases that Dr. Leyk would like us to use. TemplateDoublyLinkedLists.h contains both the definitions and the declarations for the test file mentioned earlier.

5.1) Part 2:

This used many functions from the TemplateDoublyLinkedLists.h, however since it simply segmentation faults under execution I can offer little analyzation of what the actual code would be to make it work. However, the class Record has all of the data members required, and all of the member functions. The search functions are written, and the insert orderly is written, it is simply making them all work together that is the problem.

Part 6: Instructions to Run the program

To run the program, simply navigate to the folders with all of the standard names, and type make clean then make then ./(name of main) to compile and run. The name of the folder to navigate to the book database is Book_main.

Part 7: Input and output

7.1)

SimpleDoublyLinkedList: No input and there should be several lists of numbers in various orders

DoublyLinkedList: No input and there should be several lists of numbers in various orders

TemplatedDoublyLinkedList: No input and there should be several lists of numbers in various orders

7.2)

No actual input or output, though the theoretical input should be the title, author, isbn, and output would be the book record that you want.

Part 8: Exceptions

8.1)

SimpleDoublyLinkedList: There were no exceptions to throw

DoublyLinkedList: Exceptions were implemented for the removeFirst, removeLast, last, first programs. This is because you can't interact with unlinked nodes.

TemplateDoublyLinkedList: Exceptions are implemented in the same functions for the same reasons.

8.2)

Exceptions were again implemented in the TemplateDoublyLinkedList.h for the same reasons, there was also an exception in the Book_main.cpp in the function search_through() function.

Part 9: C++ Object Oriented/Generic Programming Features

9.1)

The TemplateDoublyLinkedList is an example if generic programming used, as it can take any type.

Using inheritance and the friend class was a major generic part of the programs.

9.2)

Had it worked successfully, the range based for used to out the lists would've been a c++11 feature.

Using the TemplatedDoublyLinkedList.h was an example of generic programming.

Part 10: Testing

10.1)

The test cases were what was in each main file.

10.2)

There were no test cases.

Sources Used:

Peer TA's and the Peer TA helpdesk, cppreference, cplusplus.com, and stack overflow.