CSE 1384 - Linked Lists
Lab 7

Objectives:

- Continue practicing past concepts
- Practice using linked lists in C++

Assignment:

For this assignment, we'll be practicing using linked lists. You'll begin the assignment with some starter files (`linkedlist.cpp`) and a few files you **shouldn't change** (`linkedlist.h, node.h, main.cpp, books.txt`). The driver file (`main.cpp`) will serve as a testing file; your screenshot should match mine exactly.

The LinkedList class has some initial functions already defined:

- Constructor
- Destructor
- Display - displays the linked list
- Append - adds an item to the end of the linked list
- Pop - removes the last item from the linked list

You should not change these functions already defined. Your job will be to complete the LinkedList class to make the driver (`main.cpp`) fully functional. You can use these functions as a basis to begin creating your own functions. The linked list will be a list of books (containing two string variables: a title and an author).

A text file provided to you must be placed where the main can open it and create the initial list of books. You don't have to code the file interactions, this is done for you. Just make sure you put `books.txt` in the correct location.

The node header file (`node.h`) defines your node and your optional constructors available to you. The linked list header file (`linkedlist.h`) has definitions for the functions you should be adding to the code (in the `linkedlist.cpp`). The return values and parameters are listed appropriately there. These functions are:

- insert
    - Insert a node at the position sent. So, if position 1 is chosen, it'd be like index 1 in an array. Size should increase (AKA, the item should NOT overwrite the current)
- remove
    - Remove a node the position sent (starting at 0)
- view
    - View the book at the position sent (starting at 0)

Error Checking:

Take note that the insert, remove, and view functions have a return type of an integer. You should be returning 1 upon a failed operation and 0 upon a successful operation. Indexing starts at 0.

Hints:

Insert:

You should be able to handle the following: inserting to the head, inserting to the end of the list, inserting in the middle of the list. Consider keeping track of the item before the one you're inserting and the item in the current place you're inserting to.

Remove:

You should be able to handle the following: removing from the head, removing from the end of the list, and removing from the middle of the list. Consider keeping track of the item before the intended deleted item and the item after the deleted item. *Don't forget to explicitly delete the item using keyword "delete" to avoid memory leaks.*

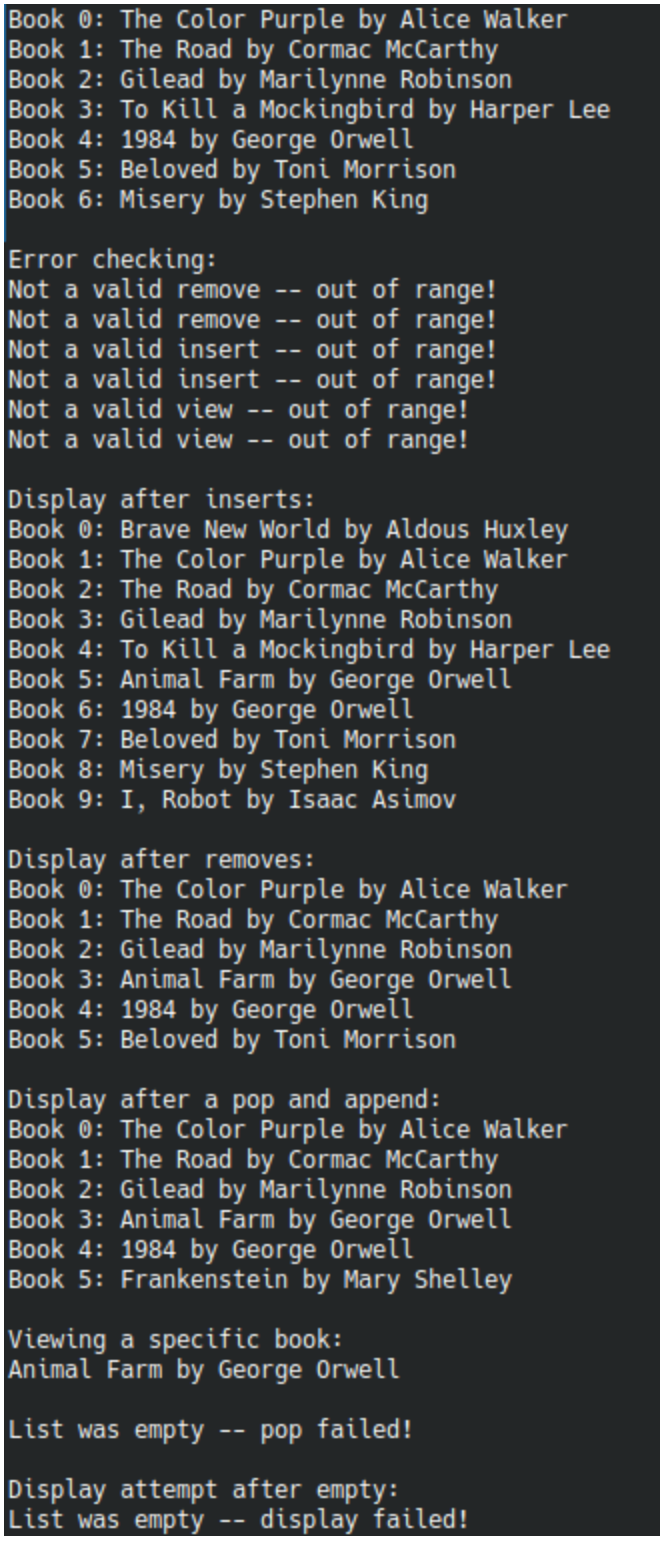
Comment Block:

Your code should contain a comment block at the top containing information on who wrote the code, what the assignment is, when it is due, etc. Here is an example of a good comment block to put:

```
/*
 Name: <your name>                      NetID: <your netID>
 Date: <current date>                   Due Date: <enter in due date>

 Description: <What is the program?>
*/
```

Execution Screenshot:



Your program (if everything is fully functional) should output this exactly.

Deliverables:

- C++ code (.cpp file)
    - If you completed everything, just submit "`linkedlist.cpp`"
    - If you didn't complete everything make sure you submit your working main.cpp as well
- A document (.pdf) with a screenshot showing the program running

Point Breakdown:
(100 points total)
A submission that doesn't contain any code will receive a 0.

Grading Note: indexing must start at zero. If it doesn't, it's not adding/removing in the correct locations and that is an incorrect add/remove.

- 10pts - pointers
    - 5pts - proper deallocation (deleting)
    - 5pts - proper allocation (creating a new instance of Node)
- 15pts - error checking
    - 5pts per function (view, insert, remove) for PROPER error checking
- 5pts - view
    - Points awarded for correctness → views correct position
- 25pts - insert
    - 10pts - head and tail insertion (5pts each)
    - 10pts - middle insertion
    - 5pts - keeps proper track of head/tail/size when necessary
- 25pts - remove
    - 10pts - head and tail removal (5pts each)
    - 10pts - middle removal
    - 5pts - keeps proper track of head/tail/size when necessary
- 10pts - execution screenshot
- 10pts - programming style *
- (penalty) -15pts - changing the code given to you (`main.cpp, node.h, linkedlist.h`, OR any of the existing functions in the `linkedlist.cpp`)

* Programming style includes good commenting, variable nomenclature, good whitespace, etc.

(Commenting out non-working portions of the main.cpp is the only acceptable change)