CSE 1384 - Sorting/Searching
Lab 9

Objectives:

- Continue practicing past concepts
- Practice using searching and sorting algorithms in C++

Assignment:

This assignment will center around handling a library of books. A text file (`books.txt`) has been provided to you. The program should handle the following items: searching through binary search (by book title), sorting descending (by book title), and a shuffle.

There's also a few things included in the class (and completed) to help you out: a constructor/destructor, display function, append, and sort ascending (by book title).

To complete the lab, all you must do is complete the functions in the class `.cpp` file given to you (`library.cpp`). That's three functions you must complete:

- int binarySearch(string title)
- void shuffle() – a strategy of YOUR DESIGN
- void reverseSort()

Notably, reverseSort should be a descending sort – you CANNOT sort the list using the algorithm already present and then reverse it. If you use the one already present, you will not get credit for this sort. There's a place in the comment to add your algorithm of choice and any links you may have used to aid you.

Do NOT change any completed code given to you.

One important thing to mention is that you are implementing searching and sorting on a LinkedList for this lab. There's no concrete indices associated with LinkedList nodes. To help you out, I included the ascending sort already – implementing select sort. Which does mean you cannot implement this sorting algorithm. But, look over it and see how it's working. I recommend using this to find out how to grab the items that you'll need for your sorts or searching.

In order to test your functions, there's a completed `main.cpp` that will allow you to do so. You shouldn't need to comment anything out to get things to work because it is setup on a menuing structure. Simply don't choose the menu option for any functions that you haven't completed yet.

You cannot use a built-in compiler sort. You must implement one on linked lists. Example: bubble sort, insertion sort, quick sort, etc. We will NOT give credit for using a function that is already included in C++.

Hint:

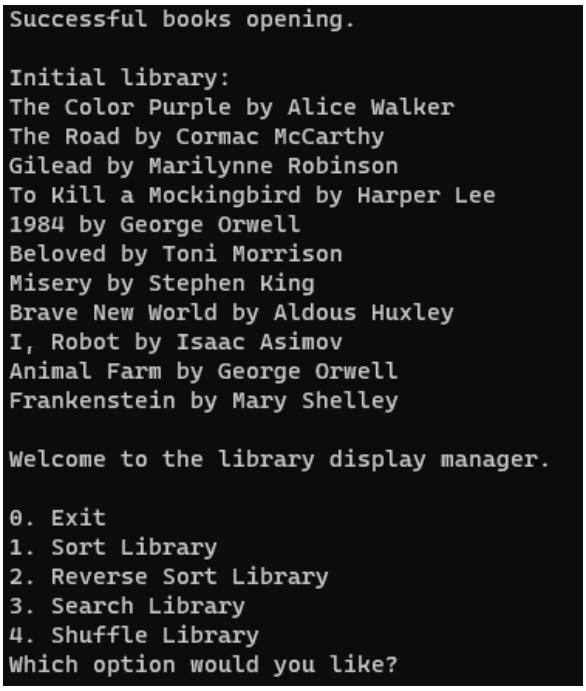
There's nothing special that needs to be done to search or sorting strings. Just use your strings in the comparison as you would integer to integer comparisons. As long as you don't accidentally mess up and compare a string to something that isn't a string, it should fit into the algorithms fine as far as data typing goes.

For the shuffle, consider using random numbers and swapping. The following libraries would be useful if using that strategy:

```
#include <time.h>
#include <stdlib.h>
```

Initial Program state:

Everyone's program should start off the same. Like so:



Option 1 should work out-of-box. Your job for the program is to make options 2, 3, and 4 work. Everything is set up to where if your files are organized correctly, you shouldn't need to do anything to get to this point.

Make sure you use what's already included to help you out when planning for the other functions. There's a lot here that can give strategies!

Milestones:

You must receive a check-off for each item to receive full credit on the lab. You may show the milestones incrementally, or receive a check off for multiple in one demo depending on your progress.

Examples of each milestone are included.

Milestone 1:

Binary search. Must show:

- Searching based on booktitle correctly
- 2 successful searches
- 1 unsuccessful search

(Please keep in mind this will NOT work properly if the list isn't sorted properly – use the sort menu option prior to the search!)

```
0. Exit
1. Sort Library
2. Reverse Sort Library
3. Search Library
4. Shuffle Library
Which option would you like? 3

What book title would you like to search? Animal Farm

Book found at position 1
```

Milestone 2:

Reverse sort. Must show:

- Successful reverse sort
- Sorting based on book title
- Student must answer to TA:
    - What sorting algorithm did you use? Why?

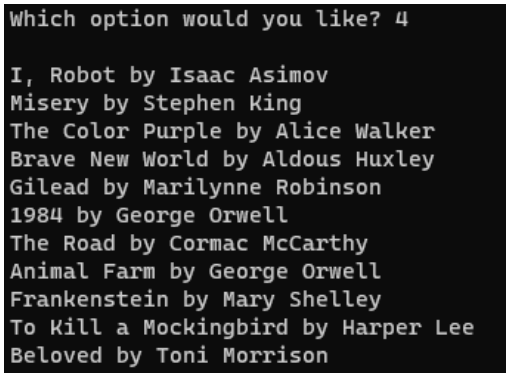
```
Which option would you like? 2

To Kill a Mockingbird by Harper Lee
The Road by Cormac McCarthy
The Color Purple by Alice Walker
Misery by Stephen King
I, Robot by Isaac Asimov
Gilead by Marilynne Robinson
Frankenstein by Mary Shelley
Brave New World by Aldous Huxley
Beloved by Toni Morrison
Animal Farm by George Orwell
1984 by George Orwell
```

Milestone 3:

Shuffle. Must show:

- Successful shuffle
- Student must answer to the TA:
    - What was your strategy while building your shuffle?
    - How does your shuffle work?

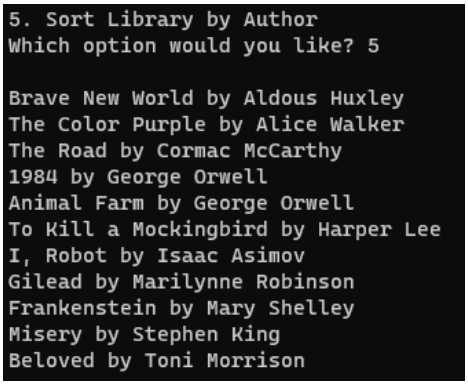


BONUS / Honors Student Credit:

If you're an honors student, you ***must*** complete this portion. If you do not, it will be counted against you. If you're not an honors student, you may complete this section for an additional 10 bonus points.

Currently, the program only sorts based on book *title.* Uncomment out the menu option in the main program to allow for sorting on book *author* and complete the coinciding function in the class files to make it work. You ***must*** use a separate sorting algorithm than those used to sort your book titles.

If there's a duplicate author, you needn't worry about sorting by title within that author.



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?>
*/
```

Deliverables:

- C++ code (`library.cpp` file)

References:

Sorting/searching algorithms are pretty set in stone. If you're using a method or strategy found on the internet in regard to a particular algorithm, you *must* reference this information.

Do **not** just put random links in your code. Tell us which part of your program was found. If you used Heap Sort in your program, tell us where you adapted the information from. If you looked up strategies to deal with strings in a Binary Search, tell us where you got it from.

<u>This is required and not optional.</u>

Point Breakdown:
(100 points total)
A submission that hasn't received a check-off will not be considered for grading.

A couple of grading notes:

Successfully applying a linear search will result in no points for milestone 1. You need to show a successful BINARY search for credit.

Using my sort and then reversing will result in no points for milestone 2. Using the same sort as my sort but changing it to be a descending sort will result in no points for milestone 2. <u>*Unique* algorithms</u> should be applied to receive credit.

Any built-in algorithms that sort/shuffle the list for you will result in no points for the associated milestone. Tools can be used in the shuffle (like I recommended), but nothing that does it for you.

Using the same sort in milestone 1 and the honors/bonus portion will result in no points for the honors/bonus portion. The honors/bonus portion using the sort used in the code provided will result in no points for the honors/bonus portion.

- 30pts - milestone 1
- 30pts - milestone 2
- 30pts - milestone 3
- 10pts - programming style *
- Honors / BONUS (dependent on Honors status):
    - Honors: 10pts penalty for not completing
    - BONUS: 10pts added for completing
- (PENALTY) 15pts - changing the code given to you (`main.cpp, node.h, library.h`, OR any of the existing functions in the `library.cpp`)
- (PENALTY) 15pts - no citation links provided
    - These don't have to be some formal citation format. You should just provide where you found your information – internet links are fine to include. Even if it just says "instructor slides" something should be provided

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