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 playlist of songs. A text file (`songs.txt`) has been provided to you. You can use this file, or create your own. If you create your own the file should still be named "`songs.txt`" to be processed correctly. The text file should contain at least **ten** songs, containing songs that begin with a large array of letters. The text file should also contain **at least one** instance of songs that begin with the same word. Note the format of the given file and follow it for your own if you elect to make your own file.

The program should handle the following items: searching through binary search (by song title), sorting ascending (by song title), and sorting descending (by song title).

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

- int binarySearch(string title)
- void sort()
- void reverseSort()

Notably, sort and reverseSort should be ascending and descending respectively and you MUST use two separate sorting algorithms to complete them. If you use the same one, you will only get credit for ONE. 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. I included a shuffle function. I recommend looking through the function to see how you may need to grab data and swap information between Nodes.

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.

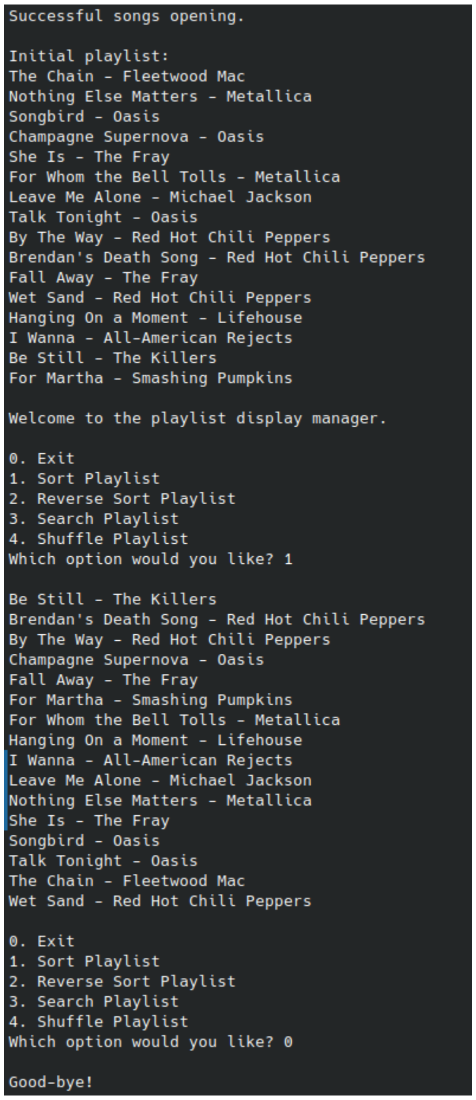
BONUS / Honors Student Credit:

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 song *title.* Uncomment out the menu option in the main program to allow for sorting on song *artist* and complete the coinciding function in the class files to make it work. You *must* use a separate sorting algorithm than those using to sort your song titles.

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

Example Execution:



```
Successful songs opening.

Initial playlist:
The Chain - Fleetwood Mac
Nothing Else Matters - Metallica
Songbird - Oasis
Champagne Supernova - Oasis
She Is - The Fray
For Whom the Bell Tolls - Metallica
Leave Me Alone - Michael Jackson
Talk Tonight - Oasis
By The Way - Red Hot Chili Peppers
Brendan's Death Song - Red Hot Chili Peppers
Fall Away - The Fray
Wet Sand - Red Hot Chili Peppers
Hanging On a Moment - Lifehouse
I Wanna - All-American Rejects
Be Still - The Killers
For Martha - Smashing Pumpkins

Welcome to the playlist display manager.

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

Be Still - The Killers
Brendan's Death Song - Red Hot Chili Peppers
By The Way - Red Hot Chili Peppers
Champagne Supernova - Oasis
Fall Away - The Fray
For Martha - Smashing Pumpkins
For Whom the Bell Tolls - Metallica
Hanging On a Moment - Lifehouse
I Wanna - All-American Rejects
Leave Me Alone - Michael Jackson
Nothing Else Matters - Metallica
She Is - The Fray
Songbird - Oasis
Talk Tonight - Oasis
The Chain - Fleetwood Mac
Wet Sand - Red Hot Chili Peppers

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

Good-bye!
```

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.

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 (`playlist.cpp` file)
- Your `songs.txt` file (important only if you used a personally made file)
- A document (.pdf) with screenshots showing the program running
  - Every option should be shown
    - Show a successful search along with an unsuccessful search
- A document (.pdf) containing the following information:
  - What sorting algorithms did you choose? Why?

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 doesn't contain any code will receive a 0.

** If the same algorithm is used for sort and reverse sort, points should only be awarded to one

- 15pts - linked list handling
  - 5pts - successfully handles navigating nodes for searching
  - 10pts - successfully handles swapping node data
- 20pts - sort (ascending)
  - 15pts - successfully sorts ascending
  - 5pts - sorts by song *title*
- 20pts - reverse sort (descending)
  - 15pts - successfully sorts descending
  - 5pts - sorts by song *title*
- 20pts - binary search
  - 15pts - successfully performs binary search
  - 5pts - searches song *titles*
- 15pts - PDF
  - 10pts - screenshots
  - 5pts - algorithm statements
- 10pts - programming style *
- Honors / BONUS (dependent on Honors status):
  - Honors: 10pts penalty for not completing
  - BONUS: 10pts added for completing
- (penalty) -15pts - changing the program given
- (penalty) -5pts - personalized songs file doesn't include at least 10 songs, songs with duplicate word beginnings (irrelevant if they used my file)

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