There is no late submission for this assignment. All assignments must be turned in by Friday April 26, 2019 (11:59 PM)

Book Class Modifications:

- Overload the output operator (<<) to print a book. The book must be printed in the following format without any new lines:
  `Title:Problem Solving with C++, Author: Walter Savitch, Published: 2018`

Write a C++ program that uses the *Book* class to search, delete, or add books. Assume that the library will contain up to 15,000 books. A data file of books will be provided. Read the books and store them into an array/vector of *Books.* The program should ignore all blank lines. Each line in the file is of the format:

```
<Title>|<Year>|<Authors>
```

The program should allow the user to do any of the following (using command line arguments). The following options should be available to the user:

Assuming your program name is "`a.out`", the command to access the library are as follows:

```
./a.out  [option] string
```

Options

`-a`	Search by author
`-t`	Search by title
`-y`	Search by year
`-n`	Add new book to the library the book must be saved to the file.
`-d`	Delete a book by title (**title must match exactly and should be case insensitive**)
`No option`	search for any string in the title, year, or author.

Examples:

```
./a.out -a "aDaMs"
./a.out -t "GALaXy"
./a.out -y "19"
./a.out -n "Problem Solving with C++|2018|Walter Savitch"
./a.out -d "Problem Solving with C++"
./a.out "C++"
```

Keep in mind that these search options might produce more than one Book.

You may use any function or library discussed in class or in the chapters we covered from your textbook. Do not use any other libraries or functions.

Parsing command line arguments

In C++ you can input data into your program on the command line (command line arguments). You can capture these data by adding two parameters to your main program as follows:

```
int main (int argc, char **argv)
```

`argc`: (argument count) number of arguments on the line, including the name of the program

`argv`: (argument vector) list of c-style strings that represent all the arguments including the name of the program (two dimensional array of characters).

For example, executing the command:

```
./a.out -a "douglas adams"
```

Assigns:

```
argc = 3

argv[0] will be "./a.out"

argv[1] will be "-a"

argv[2] will be "douglas adams"
```

Quotation is necessary if you are using white spaces within the string.

Your program must print an error message and explanation if any of the following happens:

1. Number of arguments is less than 2 or more than 3.
2. Invalid option is entered. Begins with `'-'` and is not one of the options above.
3. File access error

Hints:

- Use the third constructor of the *Book* class to initialize each book in the array/vector.
- Test your program with a small file first.
- Implement one option at a time.
- Start early.

Grading:

Programs that contain syntax errors will earn zero points.

Programs that use global variables, other than constants, will earn zero points.

Programs that do not use separate files will earn zero points.

Correctness: (34 points)

- The results must be accurate.
- All the values requested in the output are displayed as requested and are in the correct format.
  - 4 points for each of the options (24 points)
  - 5 points, loading the data into an array/vector
  - 5 points for overloading the << operator

Documentations and Style: (6 points).

Follow the coding style outline on GitHub:

https://github.com/nasseef/cs2400/blob/master/docs/coding-style.md