CSCI 1300 CS1: Starting Computing
Ashraf, Fleming, Correll, Cox, Fall 2019
Project 2: Library system
Due: Saturday, November 2nd, by 6 pm
No bonus points for early submission

Three components (Moodle CodeRunner attempts, zip file, and interview grading) must be completed and submitted by Saturday, November 2nd, 6:00 pm for your homework to receive points. Project 2 requires you to have an interview grading with your TA by November 15th

---

0. Table of Contents

---

1. Objectives

- Define classes and create objects
- Array operations: initialization, search
- Create arrays of an object type

- Use filestream objects to read data from text files

---

2. Submission Requirements

All three steps must be fully completed by the submission deadline for your homework to be graded.

1. ***Work on questions on your Cloud 9 workspace:*** You need to write your code on Cloud 9 workspace to solve questions and test your code on your Cloud 9 workspace before submitting it to Moodle. (Create a directory called **proj2** and place all your file(s) for this assignment in this directory to keep your workspace organized)

2. ***Submit to the Moodle CodeRunner:*** Head over to Moodle to the link **Project 2 CodeRunner**. You will find one programming quiz question for each problem in the assignment. Submit your solution for the first problem and press the Check button. You will see a report on how your solution passed the tests and the resulting score for the first problem. You can modify your code and re-submit (press *Check* again) as many times as you need to, up until the assignment due date. Continue with the rest of the problems.

3. ***Submit a .zip file to Moodle:*** After you have completed all questions from the Moodle assignment, zip all 7 solution files you compiled in Cloud9 (one cpp file for each problem), and submit the zip file through the **Project 2** link on Moodle.

---

3. Rubric

Aside from the points received from the **Project 2 CodeRunner** quiz problems, your TA will look at your solution files (zipped together) as submitted through the **Project 2** link on Moodle and assign points for the following:

Style, Comments, Algorithm:

Style:

- Your code should be well-styled, and we expect your code to follow some basic guidelines on whitespace, naming variables and indentation, to receive full credit. Please refer to the **CSCI 1300 Style Guide** on Moodle.

Comments:

- Your code should be well-commented. Use comments to explain what you are doing, especially if you have a complex section of code. These comments are intended to help other developers understand how your code works. These comments should begin with two backslashes (`//`) or the multi-line comments (`/* … comments here… */`) .
- Please also include a comment at the top of your solution with the following format:

```
// CS1300 Fall 2019
// Author: my name
// Recitation: 123 – Favorite TA
// Project 2 - Problem # ...
```

Algorithm:

- Before each function that you define, you should include a comment that describes the inputs and outputs of your function and what algorithms you are using inside the function.
- This is an example C++ solution. Look at the code and the algorithm description for an example of what is expected.

Example 1:

```
/*
 * Algorithm: convert money from U.S. Dollars (USD) to Euros.
 *    1. Take the value of number of dollars involved
 *       in the transaction.
 *    2. Current value of 1 USD is equal to 0.86 euros
 *    3. Multiply the number of dollars got with the
 *       currency exchange rate to get Euros value
 *    4. Return the computed Euro value
 * Input parameters: Amount in USD (double)
 * Output (prints to screen): nothing
 * Returns: Amount in Euros (double)
 */
```

Example 2:

```
double convertUSDtoEuros(double dollars)
{
     double exchange_rate = 0.86; //declaration of exchange
rate
     double euros = dollars * exchange_rate; //conversion
     return euros; //return the value in euros
}
```

The algorithm described below does not mention in detail what the algorithm does and does not mention what value the function returns. Also, the solution is not commented. This would work properly, but would not receive full credit due to the lack of documentation.

```
/*
 * conversion
 */
double convertUSDtoEuros(double dollars)
{
     double euros = dollars * 0.86;
     return euros;
}
```

4. Problem Set

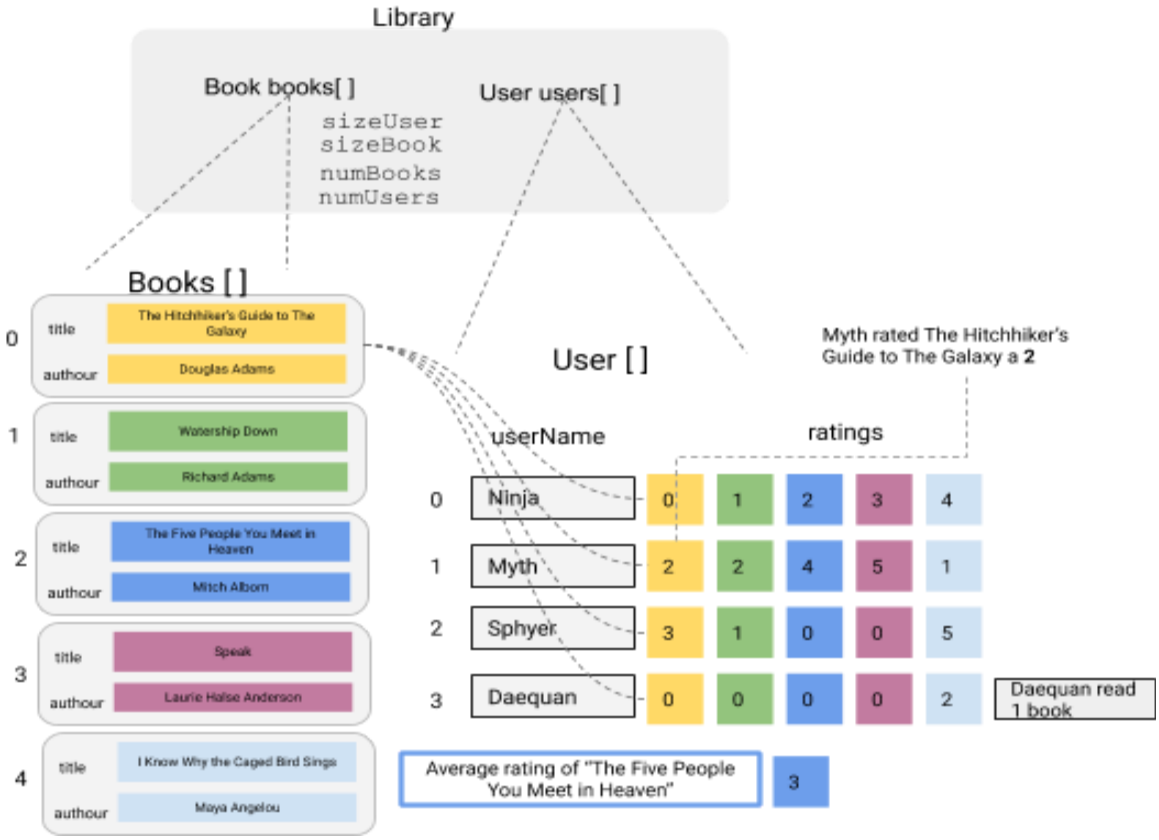
In Project 2, you will be creating a **Library** class to handle the operation of your **Book** and **User** classes from Homework 7. This new class will streamline the use of your previously-created classes, and will introduce the ability to recommend books based on the similarity between two users.

Specifications

- Create a new class **Library**. Define the class in a header file and implement it in a separate cpp file.
- The `Book` and `User` classes from Homework 7 will be part of Project 2 as well. There may be some small modifications of your member functions and class definitions to fit the new **Library** class.
- In a driver routine called `project2.cpp`, the `main()` function will create an instance of Library object and a menu as specified below.

- Students should have seven files (`Book.h`, `Book.cpp`, `User.h`, `User.cpp`, `Library.h`, `Library.cpp`, `project2.cpp`)
- The name of each member function should be exactly as specified. If you modify the function names, then your solution will not pass the autograder.
- There are two questions in Coderunner: 1) Library class 2) driver function

Visualization of various elements in Project 2



4.1. Library Class

(150 points in Coderunner)

Problem 0: `Library` Class

Create `Library.h` and `Library.cpp`, and implement a class `Library`, with separate interface and implementation, comprised of the following attributes:

Data members (private):	
int: `sizeBook`	The capacity of the `books` array (50). Constant
int: `sizeUser`	The capacity of the `users` array (100). Constant
`Book` array: `books`	An array of `Book` objects
`User` array: `users`	An array of `User` objects
int: `numBooks`	Number of books in the database (library)
int: `numUsers`	Number of users in the database (library)
Member functions (public):	
Default constructor	Sets both `numBooks` and `numUsers` to value 0.
`getSizeBook()`	Returns `sizeBook` as an integer
`getSizeUser()`	Returns `sizeUser` as an integer
`getNumBooks()`	Returns `numBooks` as an integer
`getNumUsers()`	Returns `numUsers` as an integer
`readBooks(string)`	Takes a string (the name of the file to be read) and populates the `books` array. Returns the total number of books in `books` array as an integer
`printAllBooks()`	Prints all books stored in `books` array
`printBooksByAuthor(string)`	Takes a string (the author name) and print books written by the author in `books` array
`readRatings(string)`	Takes a string (the name of the file to be read) and populates the `users` array. Returns the total number of users in `users` array as an integer
`getRating(string, string)`	Takes two strings (username and book title) and returns that user's rating for the specified book.
`getCountReadBooks(string)`	Takes a string (username) and returns the number of books read by that user as an integer.

`viewRatings(string, int)`	Takes a string (username) and a minimum rating value, and prints all the books a user has provided ratings that are greater than the value passed into the function.
`calcAvgRating(string)`	Takes a string (the title of a book) and returns the average rating of the specified book as a double
`calcAvgRatingByAuthor(string)`	Takes a string (name of author) and returns the average rating of the specified author as a double
`addUser(string)`	Takes a string (username) and returns an integer 1 if the user is successfully added, 0 if the username already exists in the users array and -2 if the users array is already full.
`checkOutBook(string, string, int)`	Takes two strings and an integer for username, title of book, and a new rating, respectively (in this order). Returns an integer 1 if the rating is successfully updated, -4 if the rating value is not valid and -3 if the rating value is valid, but the user or title does not exist in the database.
`getRecommendations(string)`	Takes a string username and prints the first 5 book recommendations from the most similar (other) user.

It is advisable to write your own test cases for each class. Test your class in Cloud9 before submitting to the autograder, because the CodeRunner autograder has a **submission limit of 30 attempts**, after which there will be a small deduction of points.

Note that the following is broken up into problems to make it a bit more digestible, and for us to break up which parts are worth which points, but there are only 4 Coderunner problems on Moodle for testing your implementations.

Problem 1: the member function `readBooks`

Update the `readBooks` function from Homework 6 to now be a member function for the `Library` class. The `readBooks` function populates an array of `Book` objects with the title and author data found in a file similar to the file `books.txt` that you've used in previous assignments. The array of `Book` objects is one of the data members of the `Library` class. This function should:

- Accept one parameter:
  - `string`: the name of the file to be read
- Use `ifstream` and `getline` to read data from the file, making an instance of the `Book` object for each line, and placing it into the `books` array.
- Return the total number of books in the system, as an integer.
- If multiple txt files are read, then the `books` array should be populated with all of the books from all of the files (unless it reaches capacity, of course). For example, suppose `readBooks` reads `books1.txt`, and then it reads `books2.txt`. After the second function call, `readBooks` returns the total number of books read from *both* files, and the `books` array stores all books from both `books1.txt` and `books2.txt`.
- The function should return the following values depending on cases:
  - Return the total number of `books` in the system, as an integer.
  - When `numBooks` is equal to the `size`, return -2.
  - When the file is not opened successfully, return -1.
  - The priority of the return code -2 is higher than -1, i.e., in cases when `numBooks` is equal to the `sizeBook` and the file cannot be opened, the function should return -2.
  - When `numBooks` is smaller than `sizeBook`, keep the existing elements in `books`, then read data from the file and add (append) the data to the array. Be sure to update the total number of books in the system. The number of books stored in the array cannot exceed the `sizeBook` of the `books` array.
- Empty lines should not be added to the arrays.

Important: Since your `books` array is private, we cannot directly check objects stored in the array form the `main()`, like you tested in Homework 7. Let's make `printAllBooks` to check if your `readBooks` are working fully functionally… in the next problem!

Example 1: `readBooks` as a general case

fileName.txt	`Author A,Book 1` `Author B,Book 2`

Function calls	`// make library object`
	`Library myLibrary`
	`// call readBooks`
	`int rv = myLibrary.readBooks("fileName.txt");`
	`// print values`
	`cout << "rv = " << rv << endl;`
	`cout << "numBooks = ";`
	`cout << myLibrary.getNumBooks() << endl;`
	
	`// print books`
	`myLibrary.printAllBooks();`

Output	`rv = 2`
	`numBooks = 2`
	`Here is a list of books`
	`Book 1 by Author A`
	`Book 2 by Author B`

Example 2: Suppose we call the `readBooks` functions twice. In `books` array, all books from the first file and the second file should be stored in the `books` array, and the function returns the total number of the books stored in the `books` array.

book1.txt	`Author A,Book 1` `Author B,Book 2`
book2.txt	`Author C,Book 3` `Author D,Book 4`

Function calls	`// make library object`
	`Library myLibrary`
	
	`// call readBooks and check return values`
	`int rv1 = myLibrary.readBooks("book1.txt");`
	`cout << "rv1 = " << rv << endl;`
	`int rv2 = myLibrary.readBooks("book2.txt");`
	`cout << "rv2 = " << rv << endl;`
	
	`// check value of getNumBooks`

	`cout << "numBooks = ";`
	`cout << myLibrary.getNumBooks() << endl;`
	
	`// print books`
	`myLibrary.printAllBooks();`

Output	`rv1 = 2`
	`rv2 = 4`
	`numBooks = 4`
	`Here is a list of books`
	`Book 1 by Author A`
	`Book 2 by Author B`
	`Book 3 by Author C`
	`Book 4 by Author D`

Example 3: file does not exist.

Function call	`// make library object`
	`Library myLibrary`
	
	`// call readBooks and check return values`
	`int rv1 = myLibrary.readBooks("badFile.txt");`
	`cout << "rv1 = " << rv << endl;`

Output	`rv1 = -1`

Example 4: `numBooks` becomes equal to the `sizeBook` and the function returns the `sizeBook`.

book1.txt	`Author A,Book 1`
	`Author B,Book 2`
	`Author C,Book 3`

Function call	`// make library obj`
	`Library myLibrary`
	
	`// multiple files were read`
	
	`// check value of getNumBooks`
	`cout << "numBooks = ";`
	`cout << myLibrary.getNumBooks() << endl;`
	
	`// call readBooks and check return values`
	`int rv1 = myLibrary.readBooks("book1.txt");`

	`cout << "rv1 = " << rv << endl;`
	
	`// check value of getNumBooks`
	`cout << "numBooks = ";`
	`cout << myLibrary.getNumBooks() << endl;`
	
	`// print books`
	`myLibrary.printAllBooks();`

Output	`numBooks = 48`
	`rv1 = 50`
	`numBooks = 50`
	`Here is a list of books`
	`(48 other books....)`
	`Book 1 by Author A`
	`Book 2 by Author B`

Example 5: `numBooks` is equal to the `sizeBook` means that the array is already full and it returns -2.

fileName.txt	`Author A,Book 1`
	`Author B,Book 2`
	`Author C,Book 3`

Function call	`// make library obj`
	`Library myLibrary;`
	
	`// multiple files were read`
	
	`// check value of getNumBooks`
	`cout << "numBooks = ";`
	`cout << myLibrary.getNumBooks() << endl;`
	
	`// call readBooks and check return values`
	`int rv1 = myLibrary.readBooks("book1.txt");`
	`cout << "rv1 = " << rv << endl;`
	
	`// check value of getNumBooks`
	`cout << "numBooks = ";`
	`cout << myLibrary.getNumBooks() << endl;`

Output	`numBooks = 50` `rv1 = -2`

```
numBooks = 50
```

Problem 2: the member function `printAllBooks`

The **printAllBooks** function from Homework 6 was very useful. We can use this function to test your `readBooks` function. So let's do it again. Write a *new* **printAllBooks** function, which will be a member function of the `Library` class, and will be useful in displaying the contents of your library.

- This function should not take any parameters.
- This function does **not** return anything.
- If the number of books is 0 or less than 0, print `"No books are stored"`.
- This function should print `"Here is a list of books"` and then each book in a new line using the following statement

```
cout << books[i].getTitle() << " by ";
cout << books[i].getAuthor() << endl;
```

Note: In the test case, you can always assume that the number of books matches the number of elements in the `books` array.

Expected output (assuming you have read the data from `books.txt`)

```
Here is a list of books
The Hitchhiker's Guide To The Galaxy by Douglas Adams
Watership Down by Richard Adams
The Five People You Meet in Heaven by Mitch Albom
Speak by Laurie Halse Anderson
...
```

Problem 3: the member function `printBooksByAuthor`

Update the `printBooksByAuthor` function from Homework 6 to now be a member function of the `Library` class. The `printBooksByAuthor` function prints the list of books that are written by the given author.

- This function should take one parameter:
  - **string:** `author name`
- You function should **not return** anything.
- If the number of books is 0 or less than 0, print `"No books are stored"`

- If there are no books by the author then you should print the following, "`There are no books by <author>`"
- If there are one or multiple books by the same author, you should print the following statement, "`Here is a list of books by <author>`" followed by, each book's title by this author in a new line.

Note: In the test case, you can always assume that the number of books matches the number of elements in the `books` array.

Example 1: There are two books by `Author A`.

bookFile.txt	`AuthorA,Book1`
	`AuthorB,Book2`
	`AuthorA,Book3`

Function Call:	`// make library obj`
	`Library lib;`
	
	`// read book file`
	`lib.readBooks("bookFile.txt");`
	
	`// call printBooksByAuthor`
	`string author = "AuthorA";`
	`lib.printBooksByAuthor(author);`

Output:	`Here is a list of books by AuthorA`
	`Book 1`
	`Book 3`

Example 2: There are no books by the `Author A`.

bookFile.txt	`AuthorA,Book1`
	`AuthorB,Book2`
	`AuthorA,Book3`

Function Call:	`// make library obj`
	`Library lib;`
	
	`// read book file`
	`lib.readBooks("bookFile.txt");`
	
	`// call printBooksByAuthor`
	`string author = "AuthorC";`
	`lib.printBooksByAuthor(author);`

Output:	`There are no books by AuthorC`

Problem 4: the member function `readRatings`

Update the `readRatings` function from Homework 7 to now be a member function for the `Library` class. The `readRatings` function populates an array of `User` objects with the username and ratings data found in a file similar to the file `ratings.txt` that you've used in previous assignments. Each username is followed by a list of ratings of the user for each book in `books.txt`. The array of `User` objects is one of the data members of the `Library` class.

For example, suppose there are a total of 3 books. The `ratings.txt` file would be of the format:

`ratings.txt`

`ritchie,3,3,3`
`stroustrup,0,4,5`
`gosling,2,2,3`
`rossum,5,5,5`
`...`

This function should:

- Accept one parameter:
  - `string`: the name of the file to be read
- Use `ifstream` and `getline` to read data from the file, making an instance of a `User` object for each line, and placing it in the `users` array.
- **Hint**: You can use the `split()` - function from homework 4 with comma (",") as the delimiter.
- You can use `stoi` to convert each rating value (a string, as read from the text file) into an integer value.
- If multiple txt files are read, then the `users` array should be populated with all of the user data from all of the files (unless it reaches capacity, of course). For example, suppose `readRatings` reads `ratings1.txt`, and then it reads `ratings2.txt`. After the second function call, `readRatings` returns the total number of users read from *both* files, and the `users` array stores all users from *both* `ratings1.txt` and `ratings2.txt`

- The function should return the following values depending on cases:
    - Case1: When `numUsers` is greater than or equal to the `sizeUser`, return -2.
    - Case2: If the file cannot be opened, return -1.
    - Case3: When `numUsers` is smaller than the `size` of `users` array, keep the existing elements in `users` array, then read data from file and add (append) the data to the arrays. The number of users stored in the arrays cannot exceed the size of the `users` array.
    - Case4: Return the total number of users in the system, as an integer.
    - Your function must check these cases in the order specified above.
    - When `numUsers` is smaller than the `sizeUser` of `users` array, keep the existing elements in `users` array, then read data from file and add (append) the data to the arrays. Be sure to update the total number of users in the system. The number of users stored in the arrays cannot exceed the size of the `users` array.
- Empty lines should not be added to the arrays.

Important: Since your `users` array is private, we cannot directly check objects stored in the array from the `main()`, like you tested in Homework 7. Let's make `getRating` (in the next problem!) to check if your `readRatings` are working well.

Example 1: `readRatings` as a general case

bookFile.txt	`AuthorA,Book1`
	`AuthorB,Book2`
	`AuthorC,Book3`
	`AuthorD,Book4`
	`AuthorF,Book5`

ratingFile.txt	`Ninja,0,1,2,3,4`
	`Myth,2,2,4,5,1`
	`Sphyer,3,1,0,0,5`
	`Daequan,0,0,0,0,2`

Function call	`// make library obj`
	`Library lib;`
	
	`// read book file`
	`lib.readBooks("bookFile.txt");`

	`// call readRatings and check return values`
	`int rv1 = lib.readRatings("ratingFile.txt");`
	`cout << "rv1 = ";`
	`cout << rv1 << endl;`
	
	`// check value of getNumUsers`
	`cout << "numUsers = ";`
	`cout << lib.getNumUsers() << endl;`
	
	`// print user's ratings`
	`string name = "Ninja"`
	`cout << lib.getRating(name, "book1") << endl;`
	`cout << lib.getRating(name, "book2") << endl;`
	`cout << lib.getRating(name, "book3") << endl;`
	`cout << lib.getRating(name, "book4") << endl;`
	`cout << lib.getRating(name, "book5") << endl;`

Output	`rv1 = 4`
	`numUsers = 4`
	`0`
	`1`
	`2`
	`3`
	`4`

Example 2: Suppose we call the `readRatings` functions twice. In `users` array, all users from the first file and the second file should be stored in the `users` array, and the function returns the total number of the users stored in the `users` array.

ratingFile.txt	`Ninja,0,1,2,3,4`
	`Myth,2,2,4,5,1`
	`Sphyer,3,1,0,0,5`
	`Daequan,0,0,0,0,2`

ratingFile2.txt	`alpha,0,1,2,3,4`
	`Beta,1,2,3,4,0`
	`gamma,3,4,0,1,2`
	`delta,2,3,4,0,1`
	`sigma,4,0,1,2,3`

Function calls	`// make library obj` `Library lib`

	`// call readRatings and check return values`
	`int rv1 = lib.readRatings("ratingFile.txt");`
	`cout << "rv1 = " << rv << endl;`
	`int rv2 = lib.readRatings("ratingFile2.txt");`
	`cout << "rv2 = " << rv << endl;`
	
	`// check value of getNumBooks`
	`cout << "numUsers = ";`
	`cout << lib.getNumUsers() << endl;`

Output	`rv1 = 4`
	`rv2 = 9`
	`numUsers = 9`

Example 3: file does not exist.

Function call	`// make library obj`
	`Library myLibrary`
	
	`// call readBooks and check return values`
	`int rv1 = myLibrary.readRatings("badFile.txt");`
	`cout << "rv1 = " << rv << endl;`

Output	`rv1 = -1`

Example 4: `numUsers` equals `sizeBook` means the array is already full.

ratingFile.txt	`Ninja,0,1,2,3,4`
	`Myth,2,2,4,5,1`
	`Sphyer,3,1,0,0,5`
	`Daequan,0,0,0,0,2`

Function call	`// make library obj`
	`Library myLibrary`
	
	`// multiple files were read`
	
	`// check value of getNumUsers`
	`cout << "numUsers = ";`
	`cout << myLibrary.getNumUsers() << endl;`
	
	`// call readRatings and check return values`

	`int rv1 = myLibrary.readRatings("ratingFile.txt");`
	`cout << "rv1 = " << rv1 << endl;`
	
	`// check value of getNumUsers`
	`cout << "numUsers = ";`
	`cout << myLibrary.getNumUsers() << endl;`
	
	`// call readRatings again`
	`int rv2 = myLibrary.readRatings("ratingFile.txt");`
	`cout << "rv2 = " << rv2 << endl;`

Output	`numUsers = 98`
	`rv1 = 100`
	`numUsers = 100`
	`rv2 = -2`

Problem 5: the member function `getRating`

The member function `getRating` accepts the given user's name and a book's title, and returns the rating that the user gave for that book.

- Your function MUST be named `getRating`.
- Your function should take 2 parameters in the following order:
  - `string`: username
  - `string`: title of the book
- The username and book title search should be case insensitive. For example, "Ben, "ben" and "BEN" are one and the same user.
- If both the user name and the book title are found, then the function should return the user's rating value for that book title.
- The function should return the following values depending on cases:
  - Return the rating value if both user and title are found
  - Return -3 if either the user or the title are not found

Set up for the examples below

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,1,4,2` `User2,0,5,3`

	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users to Library`
	`myLibrary.readRatings("ratingFile.txt");`

Example 1: Both the `userName` and `bookTitle` exists, and the value of rating is non-zero, returns the value of the given user's rating for the given book

Function call	`getRating("User1", "Title2");`
Return value	`4`

Example 2: The `userName` does not exist, it returns - 3

Function call	`getRating("User4", "Title1");`
Return value	`-3`

Example 3: The `bookTitle` does not exist, it returns - 3

Function call	`getRating("User1", "Title10");`
Return value	`-3`

Example 4: The `userName` and the `bookTitle` do not exist, returns -3

Function call	`getRating("User12", "Title10");`
Return value	`-3`

Problem 6: the member function `getCountReadBooks`

The member function `getCountReadBooks` which determines how many books a particular user has read and reviewed. This function should:

- Accept one parameter:
  - `string`: username
- The username and book title search should be case insensitive. For example, “Ben, “ben” and “BEN” are one and the same user.
- The function should return the following values depending on cases:
  - Return the number of books read/reviewed by the specified user if user is found
  - Return -3 if the username is not found or there is no book.

Example 1: The library is initialized

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks(“bookFile.txt”);`
	
	`//add users`
	`myLibrary.readRatings(“ratingFile.txt”);`
	
	`// viewRatings for User2`
	`cout << myLibrary.getCountReadBooks(“User2”);`

outputs	`2`

Example 2: The user does not exist

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users`
	`myLibrary.readRatings("ratingFile.txt");`
	
	`cout << myLibrary.getCountReadBooks("User4");`

outputs	`-3`

Example 3: The user has not rated any book yet

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users`
	`myLibrary.readRatings("ratingFile.txt");`
	
	`// getCountReadBooks for User3`
	`cout << myLibrary.getCountReadBooks("User3");`

outputs	`0`

Problem 7: the member function `viewRatings`

Create a new member function `viewRatings` prints all the books a user has provided ratings for, if the rating is greater than or equal to a minimum rating value. Recall that a rating a book 0 means a user has not rated that book and hence shouldn't be displayed. This function should:

- Accept one parameter:
  - `string`: username
  - `int`: min rating value
- Not return anything.
- If the user is not found in the database, print:

  ```
  <username> does not exist.
  ```

- If the user is found in the database, but has not rated any books with the minimum rating, print:

  ```
  <username> has not rated any books yet.
  ```

- If the user exists in the database, and has rated at least one book with a rating greater than or equal to the minimum rating provided, display the user's ratings in the following format:

Expected output (assuming you have read the data **only** from `books.txt,ratings.txt`)

`Here are the books that megan rated`
`Title : The Hitchhiker's Guide To The Galaxy`
`Rating : 5`
`-----`
`Title : The Five People You Meet in Heaven`
`Rating : 2`
`-----`
`(...)`

Example 1: The library is initialized and minimum rating is 2

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users`
	`myLibrary.readRatings("ratingFile.txt");`
	
	`myLibrary.viewRatings("User1", 2);`

outputs	`Here are the books that User2 rated`
	`Title : Title2`
	`Rating : 4`
	`-----`
	`Title : Title3`
	`Rating : 2`
	`-----`

Example 2: The user has not rated any book yet

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users`
	`myLibrary.readRatings("ratingFile.txt");`
	
	`myLibrary.viewRatings("User3", 1);`

outputs	`User3 has not rated any books yet.`

Example 3: The user has not rated any book with the minimum rating

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,1,1,1`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users`
	`myLibrary.readRatings("ratingFile.txt");`
	
	`myLibrary.viewRatings("User3", 2);`

outputs	`User3 has not rated any books yet.`

Example 4: The user does not exist

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users`
	`myLibrary.readRatings("ratingFile.txt");`

	`myLibrary.viewRatings("User4", 3);`
outputs	`User4 does not exist.`

Problem 8: the member function `calcAvgRating`

The member function `calcAvgRating` returns the average (mean) rating for a particular book. This function should:

- Accept one parameter:
  - `string`: book title
- The book title search should be case insensitive. For example, "InkHeart", "inkheart" and "INKHEART" refer to the same book.
- The average rating is calculated by the sum of non-zero ratings (for that book) divided by the number of non-zero ratings (for that book).
- The function should return the following values depending on cases:
  - Return the average rating of the specified book as a `double` if title is found
  - Return -3 if the title is not found or there are no users
  - Return 0 if the book has not been read by anyone. Poor book!

Note: Books that haven't been read (have a rating value of 0) ***shouldn't*** *be counted in calculating the average.*

Example 1: The library is initialized, and we can calculate an average ratings for the book.

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`

	`//add users`
	`myLibrary.readRatings("ratingFile.txt");`
	
	`// calcAvgRating for Title2`
	`cout << myLibrary.calcAvgRating("title2");`

Output	`4.5`

The function returns 4.5 because User1 gave Title2 a rating of 4 and User2 gave Title2 a rating of 5. The average is (4 + 5) / 2 = 4.5. Since User3's rating is 0, it is not included in the calculation.

Example 2: The title does not exist in the library. No user has read a particular title

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,0,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users`
	`myLibrary.readRatings("ratingFile.txt");`
	
	`// calcAvgRating for Title4`
	`cout << myLibrary.calcAvgRating("Title4");`
	
	`// calcAvgRating for Title1`
	`cout << myLibrary.calcAvgRating("Title1");`

Output	`-3` `0`

Problem 9: the member function `calcAvgRatingByAuthor`

The member function `calcAvgRatingByAuthor` returns the average (mean) rating for a particular author, i.e. it returns the average over all the ratings of all the books written by that particular author. This function should:

- Accept one argument:
  - `string`: author name
- The author name search should be case insensitive. For example, “Ben”, “ben” and “BEN” refer to the same author.
- The average rating is calculated by the sum of non-zero ratings for all books by that author, divided by the number of non-zero ratings (for all books by that author).
- The function should return the following values depending on cases:
  - Return the average rating of the specified author as a `double` if the author is found
  - Return -3 if the author is not found or there are no users
  - Return 0 if none of the books by the author have been read by any user. Poor author!

Note: Books that haven’t been read (have a rating value of 0) ***shouldn’t*** *be counted in calculating the average.*

Example 1: The library is initialized, and we can calculate the average ratings for a particular author.

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`
	`Author1,Title4`
	`Author2,Title5`
	`Author1,Title6`

ratingFile.txt	`User1,1,4,2,0,1,5`
	`User2,0,5,3,4,2,0`
	`User3,0,0,0,3,2,1`

Function calls	`//Create a new Library` `Library myLibrary;`

	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users`
	`myLibrary.readRatings("ratingFile.txt");`
	
	`// calcAvgRatingByAuthor for Author1`
	`cout << myLibrary.calcAvgRatingByAuthor("author1");`

Output	`2.8`

Author1 has written three books: Title1, Title4 and Title6. The columns corresponding to these titles in ratingsFile.txt are in bold:

User1,**1**,4,2,**0**,1,**5**
User2,**0**,5,3,**4**,2,**0**
User3,**0**,0,0,**3**,2,**1**

The average rating for the author is the sum of all non-zero ratings in bold, divided by the number of those non-zero ratings. Average rating = (1+4+3+5+1) / 5 = 14/5 = 2.8
User2's rating of Title1 and Title6 is 0, and User3's rating of Title1 is 0. These are not included in the calculation.

Example 2: The author does not exist in the library. No user has read any book by the author.

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`
	`Author1,Title4`
	`Author2,Title5`
	`Author1,Title6`

ratingFile.txt	`User1,1,0,2,0,0,5`
	`User2,0,0,3,4,0,0`
	`User3,0,0,0,3,0,1`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users`

	`myLibrary.readRatings("ratingFile.txt");`
	
	`// calcAvgRatingByAuthor for Author4`
	`cout << myLibrary.calcAvgRatingByAuthor("author4");`
	
	`// calcAvgRatingByAuthor for Author2`
	`cout << myLibrary.calcAvgRatingByAuthor("Author2");`

Output	`-3` `0`

Problem 10: the member function `addUser`

The member function `addUser` adds a new user to the database. This function should:

- Accept one parameter:
  - `string`: user name
- The user name is case insensitive (i.e. "Ben", "ben" and "BEN" refer to the same user).
- Fill in the `username` and `ratings` data members for a `User` object, at the first unused position in the array of `User` objects.
- Be sure to update the total number of users in the system
- The function returns following one of the following integer values:
  - Return 1 if the user is successfully added.
  - Return 0 if the username already exists in the `users` array.
  - Return -2 if the `users` array is already full.

Example 1: A user is successfully added to the `users` array.

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`myLibrary.readRatings("ratingFile.txt");`
	
	`// checking the user count`

	`cout << "numUsers = " << myLibrary.getNumUsers() << endl;`
	
	`//add users`
	`cout << myLibrary.addUser("User4");`
	
	`// checking the user count`
	`cout << "numUsers = " << myLibrary.getNumUsers() << endl;`

Output	`numUsers = 3`
	`1`
	`numUsers = 4`

Note that at this point, there are 4 users in the system, *and User3 and User4 both have all 0 ratings associated with them.*

Example 2: The username already exists in the users array (and is case-insensitive)

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add users`
	`myLibrary.readRatings("ratingFile.txt");`
	
	`//add users`
	`cout << myLibrary.addUser("user2");`

Output	`0`

Example 3: The `users` array is full

Function calls	`//Create a new Library`
	`Library myLibrary;`
	
	`//add books to Library`
	`myLibrary.readBooks("bookFile.txt");`
	
	`//add users`

	`myLibrary.readRatings("100UsersFile.txt");`
	
	`// check value of getNumBooks`
	`cout << "numUsers = ";`
	`cout << myLibrary.getNumUsers() << endl;`
	
	`// add users`
	`cout << myLibrary.addUser("user4");`

Output	`numUsers = 100` `-2`

Problem 11: the member function `checkOutBook`

The member function `checkOutBook` updates the rating of a particular book, for a particular user. This function should:

- Accept three parameters in this order
  - `string`: username
  - `string`: book title
  - `int`: new rating
- Find the index of the user and the index for the book, then update the new rating if the new rating value is valid. The rating scheme follows the one provided in homework 6.

Rating	**Meaning**
0	Did not read
1	Hell No - hate it!!
2	Don't like it.
3	Meh - neither hot nor cold
4	Liked it!
5	Mind Blown - Loved it!

- The username and book title search should be case insensitive. For example, "Ben", "ben" and "BEN" refer to the same user.

- The function returns the following integer value depending on the case (with the following order of precedence):
    - Return 1 if the rating is successfully updated
    - Return -4 if the rating value is not valid
    - If the rating value is valid, but the user or title do not exist in the database, this function should return -3.

Set up for the examples below:

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`

ratingFile.txt	`User1,1,4,2`
	`User2,0,5,3`
	`User3,0,0,0`

Set up	`//Create a new Library`
	`Library myLib;`
	
	`myLib.readBooks("bookFile.txt");`
	`myLib.readRatings("ratingFile.txt");`

Example 1: User successfully checks out a book and updates an existing rating.

Function calls	`int oldRating = myLib.getRating("User2", "Title1");`
	
	`int rv = myLib.checkOutBook("User2", "Title1", 2);`
	
	`int newRating = myLib.getRating("User2", "Title1");`
	
	`cout << "rv = " << rv << endl;`
	`cout << "oldRating = " << oldRating << endl;`
	`cout << "newRating = " << newRating << endl;`

Output	`rv = 1`
	`oldRating = 0`
	`newRating = 2`

Example 2: The rating value is invalid.

Function calls	`int oldRating = myLib.getRating("User2", "Title1");`
	
	`int rv = myLib.checkOutBook("User2", "Title1", 10);`
	
	`int newRating = myLib.getRating("User2", "Title1");`
	
	`cout << "rv = " << rv << endl;`
	`cout << "oldRating = " << oldRating << endl;`
	`cout << "newRating = " << newRating << endl;`

Output	`rv = -4`
	`oldRating = 0`
	`newRating = 0`

Since the rating value is invalid, User2's rating of Title1 should stay the same.

Example 3: A book with the given title is not found.

Function calls	`int rv = myLib.checkOutBook("User2", "noTitle", 1);`
	
	`cout << "rv = " << rv << endl;`

Output	`rv = -3`

Example 4: A user with the given name is not found.

Function calls	`int rv = myLib.checkOutBook("noUser", "title1", 2);` `cout << "rv = " << rv << endl;`
Output	`rv = -3`

Problem 12: the member function `getRecommendations`

The member function `getRecommendations` will recommend book titles a user might enjoy, based on book ratings of another user who likes similar books. This function should:

- Accept one parameter:
    - `string`: username

- Not return anything.
- Find the user with the given username, and print some book recommendations to the screen. (Details on how to recommend books are given below.)
- The username search should be case insensitive. For example, “Ben”, “ben” and “BEN” refer to the same user.
- If the user name is not found, it should print the following message :

```
<username> does not exist.
```

- If there are no books to recommend for the user, it should print the following message:

```
There are no recommendations for <username> at present.
```

- If there is at least one book to recommend for a certain user, print the following information for at most five books:

```
Here is the list of recommendations
<book_title_1> by <author1>
<book_title_2> by <author2>
…
…
<book_title_5> by <author5>
```

How to find books to recommend?

The recommendations for a given user will be based on another user who is most similar to that user. To generate recommendations, for example, for a user named Ben:

1. Find the most similar user to Ben. Let’s say we found Claire to be most similar.
2. Recommend to Ben the first 5 books in the database Claire has rated with a rating of 3, 4 or 5, that Ben has not yet read (rating 0).
3. If there are fewer than 5 books to recommend, recommend as many as possible. Ben will be presented with between 0 and 5 recommendations.

In order to compare two users and calculate their similarity, we will be looking at the rating values for all the books for **both** users (regardless of whether a user has read a book or not), and calculating the difference in their ratings. Because our similarity metric is based on difference, more similar users will have ***smaller similarity values***. Therefore,

when Ben is compared to all other users in the database, the user whose similarity score with Ben is ***smallest*** will be the most similar user (Claire).

Note 1: A new user, who has not rated any books, cannot be chosen as the most similar user. The `getCountReadBooks` function can be used to weed out the new users.

Note 2: In the event of a tie between two users for being the most similar to the user you are making recommendations for, make recommendations using the user with the *lower* index within the users array.

The similarity metric you should use is the **sum of squared differences (SSD)**. The **sum of squared differences** is calculated by summing the squares of the differences between corresponding elements in two ratings arrays from two users. Follow the example below.

Let *A* represent ben's ratings, and *B* represent claire's ratings.
A_i is ben's rating for book *i*, and B_i is claire's rating for book *i*

$$SSD = \sum_i (A_i - B_i)^2$$

Example 1 : Calculating SSD

$john's\ ratings: [0, 1, 3, 5]$
$claire's\ ratings: [3, 0, 5, 0]$

$$SSD = (0-3)^2 + (1-0)^2 + (3-5)^2 + (5-0)^2$$
$$SSD = (-3)^2 + (1)^2 + (-2)^2 + (5)^2$$
$$SSD = 9 + 1 + 4 + 25 = 39$$

Example 2: Users with very different ratings will get a high SSD.

$john's\ ratings: [5, 1, 0, 0, 5]$
$david's\ ratings: [1, 5, 0, 5, 1]$

$$SSD = (5-1)^2 + (1-5)^2 + (5-0)^2 + (5-1)^2$$
$$SSD = 4^2 + 4^2 + 5^2 + 4^2$$
$$SSD = 16 + 16 + 25 + 16 = 73$$

Example 3: Two users with very similar ratings will get a low SSD.

$$\textit{john's ratings}: [5, 0, 5, 3]$$
$$\textit{claire's ratings}: [5, 0, 4, 2]$$

$$SSD = (5-5)^2 + (5-4)^2 + (3-2)^2$$
$$SSD = 0^2 + 1^2 + 1^2$$
$$SSD = 0 + 1 + 1 = 2$$

For example (this example is different than the data in ratings.txt):
Let's say we're generating recommendations for John. Here are the books:

```
Douglas Adams,The Hitchhiker's Guide To The Galaxy
Richard Adams,Watership Down
Mitch Albom,The Five People You Meet in Heaven
Laurie Halse Anderson,Speak
```

```
Liz:   [5, 1, 5, 3]
John:  [5, 0, 3, 0]
David: [4, 1, 0, 5]
```

To generate recommendations for John:

1. **Find the most similar user**
   John has a SSD of 14 with Liz, and an SSD of 36 with David, so John is more similar to Liz. Thus, our book recommendations will be based on Liz's ratings.

2. **Find 5 books Liz (the most similar user) has rated as a 3, 4, or 5 that John has not yet read (rating 0)**
   We look at Liz's ratings to find books that she has rated that John has not:
   - Liz has rated `The Hitchhiker's Guide To The Galaxy` as 5, but John has already rated this book.
   - Liz has rated `Watership Down` as 1. John hasn't read that book yet, but the rating value of 1 is lower than 3, so we **do not** add it to the list of recommendations.
   - Liz has rated `The Five People You Meet in Heaven` as 5, but John has already rated this book.
   - Liz has rated `Speak` as 3. John has not yet read that book, so we add it to the list of recommendations.

- There are no more books that Liz has rated, so we're done. Our final list of recommendations will be:

  ```
  Speak by Laurie Halse Anderson
  ```

Set-up for the examples:

bookFile.txt	`Author1,Title1`
	`Author2,Title2`
	`Author3,Title3`
	`Author4,Title4`
	`Author5,Title5`

ratingFile.txt	`User1,5,4,2,3,1`
	`User2,5,5,3,2,0`
	`User3,0,0,0,0,0`
	`User4,0,0,0,0,0`
	`User5,5,0,2,3,0`

Function calls	`//Create a new Library`
	`Library myLib;`
	
	`myLib.readBooks(“bookFile.txt”);`
	`myLib.readRatings(“ratingFile.txt”);`

Example 1: There are books to recommend.

Function call	`myLib.getRecommendations("User5");`
Output	`Here are the list of recommendations` `Title2 by Author2`

The best matched user for User5 is User1, with SSD = 17. Title2 is the only book to recommend because that is the only book User5 has not rated and User1 rated at least a 3.

Example 2: There are no books to recommend.

Function call	`myLib.getRecommendations("User2");`
outputs	`There are no recommendations for User2 at present`

The best matched user for User2 is User1, with SSD = 4. The only book User2 has not read is Title5, but it cannot be recommended because User2 rated it as 1. Therefore, there are no books to recommend.

Example 3: The most similar user has not rated any book. So we need to find the second most similar user

Function call	`myLib.getRecommendations("User4");`

outputs	`Here are the list of recommendations`
	`Title1 by Author1`
	`Title4 by Author4`

The SSD score between User4 and User3 is 0. However, since the User3 has not rated any books, we need to find the other most similar user. The best matched user for User4 is User5, with SSD = 38. We will recommend Title1 and Title4.

4.2. Driver

(50 points in codeRunner)

Problem 13 - Driver

Now, let's modify our **HW7.cpp** from Homework 7 to use the `Book` class, `User` class, and `Library` class we have updated/created for Project 2. Make a copy of your old HW7.cpp and rename it to **project2.cpp**. Then modify it for this problem so we can show off to our enemies all the cool stuff we're doing!

Since we've added some other functionality to our driver routine, we will need to update some of the menu functionality. Download the **Project2Template.cpp** to update your `displayMenu` function, as well as some other print statements.

The zip file submission should have five files for this problem: **Book.h**, **Book.cpp**, **User.h**, **User.cpp, Library.h**, **Library.cpp**, and a driver called **project2.cpp**, with a `main()` function to test your menu interface. Note that the submitted project2.cpp file should ***not*** have the class definitions in it. They should be contained in their respective modules (Book.h+Book.cpp, User.h+User.cpp, Library.h+Library.cpp,) and **#include**'ed in project2.cpp.

For **Coderunner**, however, paste your entire project2.cpp driver function ***with*** the class definitions. You need to submit the entire program project2.cpp, including the Book, User, and Library classes, in the answer box of the Coderunner auto-grader on Moodle.

The menu will run on a loop, continually offering the user eleven options until they opt to quit. You need to fill in the code for each of the options. You should make use of the functions you wrote previously, call them, and process the values they return.

- **Option 1:** Initialize library
  - Prompt the user for a file name.
    - `Enter a book file name:`
  - Pass the file name to your `readBooks` function.
  - Print the total number of books in the database in the following format:
    - `Total books in the database: <numberOfBooks>`
  - If the function returned -1, then print the following message:
    - `No books saved to the database.`
  - If the function returned -2, print
    - `Database is already full. No books were added.`
  - When `numBooks` is equal to size of the array print the following message:
    - `Database is full. Some books may have not been added.`
- **Option 2:** Initialize user catalog
  - Prompt the user for a file name.
    - `Enter a user file name:`
  - Pass the file name to your `readRatings` function
  - Print the total number of users in the database in the following format:
    - `Total users in the database: <numUsers>`
  - If the function returned -1, then print the following message:
    - `No users saved to the database.`
  - If the function returned -2, print
    - `Database is already full. No users were added.`
  - When `numUsers` is equal to size of the array print the following message:

        - `Database is full. Some users may have not been added.`

- **Option 3:** Display library
    - If the database has not been initialized (i.e., arrays of books and users/ratings have not yet both been read in), then print
        - `Database has not been fully initialized.`
    - Otherwise, call your `printAllBooks` function.

- **Option 4:** Display books by a particular author
    - If the database has not been initialized (i.e., arrays of books and users/ratings have not yet both been read in), then print
        - `Database has not been fully initialized.`
    - Otherwise
        - Prompt the user for an author name
            - `Enter an author name:`
        - pass the author name to your `printBooksByAuthor` function

- **Option 5:** Get a rating
    - If the database has not been initialized, print
        - `Database has not been fully initialized.`

    - Otherwise:
        - Prompt the user for a username
            - `Enter a user name:`
        - Prompt the user for a title
            - `Enter a book title:`
        - Pass the username and the title to your `getRating` function

    - If the user exists in the system, print the result in the following format:
        - `<name> rated <title> with <rating>`

    - If the function returns 0, print the result in the following format:
        - `<name> has not rated <title>`

    - If the function returns -3, print the result in the following format:
        - `<name> or <title> does not exist.`

- **Option 6:** Get number of books the user has rated

    - If the database has not been initialized, print
        - `Database has not been fully initialized.`

    - Otherwise:
        - Prompt the user for a username.
            - `Enter a user name:`
        - Pass the username to your `getCountReadBooks` function

    - If the user exists in the system and has rated some books:
        - `<name> rated <number> books.`

    - If the function returns 0, print the result in the following format:
        - `<name> has not rated any books.`

    - If the function returns -3, print the result in the following format:
        - `<name> does not exist.`

- **Option 7:** View user's ratings
    - If the database has not been initialized, print
        - `Database has not been fully initialized.`

    - Otherwise:
        - Prompt the user for a username.
            - `Enter a user name:`
        - Prompt the user for minimum rating.
            - `Enter a minimum rating:`
        - Pass the username and minrating to your `viewRatings` function

- **Option 8:** Calculate the average rating for the book
    - If the database has not been initialized, print
        - `Database has not been fully initialized.`

    - Otherwise:
        - Prompt the user for a title.
            - `Enter a book title:`
        - Pass the title to your `calcAvgRating` function

    - If the title exists in the system, display the average ratings in two decimal places:

        - `The average rating for <title> is <avg rating>`

    - If the function returns -3, print the result in the following format:
        - `<title> does not exist.`

- **Option 9:** Calculate the average rating by the author
    - If the database has not been initialized, print
        - `Database has not been fully initialized.`

    - Otherwise:
        - Prompt the user for author name.
            - `Enter an author name:`
        - Pass the author name to your `calcAvgRatingByAuthor` function

    - If the author name exists in the system, display the average ratings
        - `The average rating for <authorName> is <avg rating>`

    - If the function returns -3, print the result in the following format:
        - `<authorName> does not exist.`

- **Option 10:** Add a user to the database.
    - Prompt the user for a username.
        - `Enter a user name:`
    - Pass the username to your `addUser` function
    - If the user is successfully added (the function returns 1), the print
        - `Welcome to the library <username>`

    - If the username exists in the system (the function returns 0), print
        - `<username> already exists in the database.`

    - If the function returns -2, print the result in the following format:
        - `Database is already full. <username> was not added.`

- **Option 11:** Check out the book
    - If the database has not been initialized, print
        - `Database has not been fully initialized.`

    - Otherwise:
        - Prompt the user for a username.
            - `Enter a user name:`
        - Prompt the user for a title.
            - `Enter a book title:`
        - Prompt the user for a new rating.
            - `Enter a new rating:`
        - Pass the username, title, and rating to your `checkOutBook` function

    - If the user is successfully added (the function returns 1), the print
        - `We hope you enjoyed your book. The rating has been updated.`

    - If the function returns -4, print:
        - `<rating> is not valid.`

    - If the function returns -3, print:
        - `<name> or <title> does not exist.`

- **Option 12:** get recommendations
    - If the database has not been initialized, print
        - `Database has not been fully initialized.`

    - Otherwise:
        - Prompt the user for a username.
            - `Enter a user name:`
        - Pass the username to your `getRecommendations` function

- **Option 13:** Quit
    - Print `"Good bye!"` before exiting
- **Invalid input**
  If the user input is not the above values print `Invalid input.`

Below is an example of running the `Project2_cpp` program:

```
Select a numerical option:
======Main Menu=====
```

```
1. Read books
2. Read ratings
3. Print all books
4. Print books by author
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
13
8
Database has not been fully initialized.

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. Print all books
4. Print books by author
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
1
Enter a book file name:
books.txt
Database is full. Some books may have not been added.

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. Print all books
```

```
4. Print books by author
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
2
Enter a user file name:
ratings.txt
Total users in the database: 86

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. Print all books
4. Print books by author
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit

8
Enter book title:
Is this book?
this book does not exist.

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. Print all books
4. Print books by author
```

```
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
10
Enter username:
newUser
Welcome to the library newUser

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. Print all books
4. Print books by author
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
11
Enter username:
kimberly
Enter book title:
The Shadow Club
Enter rating for the book:
5
We hope you enjoyed your book. The rating has been updated.

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
```

```
3. Print all books
4. Print books by author
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
12
Enter a username:
adam
Here is the list of recommendations
The Lion the Witch and the Wardrobe by C S Lewis

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. Print all books
4. Print books by author
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
6
Enter a username:
Malvika
Malvika does not exist.

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. Print all books
```

```
4. Print books by author
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
7
Enter a user name:
tiffany
tiffany rated 41 books.

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. Print all books
4. Print books by author
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
8
Enter a book title:
Naruto
The average rating for Naruto is 2.67105

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. Print all books
4. Print books by author
5. Get rating
```

```
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
119
invalid input

Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. Print all books
4. Print books by author
5. Get rating
6. Find number of books user rated
7. View ratings
8. Get average rating
9. Get average rating by author
10. Add a user
11. Checkout a book
12. Get recommendations
13. Quit
13
good bye!
```

6. Project 2 checklist

Here is a checklist for submitting the assignment:

1. Complete the code **Project 2 CodeRunner**
2. Submit one zip file to Project 2. The zip file should be named, **proj2_lastname.zip**. It should have the following 7 files:
    - Book.h
    - Book.cpp
    - User.h
    - User.cpp

    - Library.h
    - Library.cpp
    - Project2.cpp
3. Submit the zip file to **Project 2**
4. Sign up for the interview grading slot on Moodle. Please make sure that you sign-up and complete an interview grading with your TA by November 15th. The schedulers for interview grading will be available after the deadline of this project.
5. You are allowed to reschedule without any penalty only one time during the semester. If you miss a second time, you will be allowed to reschedule but you will incur a 25 points (out of 100) penalty, then 50 points for the third time.

7. Project 2 points summary

Criteria	**Pts**
CodeRunner	200
Interview Grading (including comments, style, algorithms)	80
Recitation attendance (week 9)*	-30
Total	280

* if your attendance is not recorded, you will lose points. Make sure your attendance is recorded on Moodle.