

CSCI 1300 CS1: Starting Computing  
Instructor: Fleming/Wong, Spring 2019  
Project 2  
Due Saturday, March 23, by 6 pm

All 3 components (Cloud9 workspace, Moodle Coderunner attempts, and zip file) must be completed and submitted by Saturday, March 23, at 6 pm for your homework to receive points.

Project 2 requires you to have an interview grading with your TA, completed by Monday, April 15 (Tax Day).

---

## 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. **Create Project2 directory on your Cloud 9 workspace:** Your recitation TA will review your code by going to your Cloud9 workspace. *TAs will check the last version that was saved before the submission deadline.*
  - Create a directory called **Project2** and place all your file(s) for this assignment in this directory.
  - Make sure to save the final version of your code (File > Save). Verify that this version displays correctly by going to File > File Version History.
  - The file(s) should have all of your functions, test cases for the functions in `main()` function(s), and adhere to the style guide. Please read the [submission file instructions](#) under Week 4. **You must include a test case for each one of your member functions for your classes.**

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 9 questions from the Moodle assignment, zip all 15 files you compiled in Cloud9, and submit the zip file through the [Project 2 \(File Submission\)](#) 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 \(File Submission\)](#) link on Moodle and assign points for the following:

#### **Style and Comments (5 points):**

- The [style guide](#) is posted on Moodle under Week 6.
- Your code should be well-commented. Please review the standard for well-commented code, presented in more detail in previous homework write-ups.
- Please also include a comment at the top of your solution with the following format:

```
// CS1300 Spring 2019
// Author: my name
// Recitation: 123 - Favorite TA
// Cloud9 Workspace Editor Link: https://ide.c9.io/...
// Project 2 - Problem # ...
```

#### **Global variables (use will result in a 5 point deduction):**

- Later in the semester, we will learn about global variables and the joys and dangerous therein. To keep things simple, straightforward, and easy to debug and test, **you may not use global variables in this homework.**

### **Algorithm (5 points):**

- 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. Please review the standard for including your algorithm for each function, presented in more detail in previous homework write-ups.

### **Test Cases:**

- Test cases for Project 2 are not a part of the grade. However, we expect for you to write your own test cases or use examples in this write-up to test (and debug) your code. In the Coderunner, you are allowed to use up to 30 times to click “check” button without penalty. After 30 checks, points will be deducted.

Please make sure that your submission files follow the the [submission file instructions](#) under Week 6.

---

## **4. Problem Set**

*\*All the examples and values used in examples are arbitrary and randomly generated.\**

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):	
<code>int: sizeBook</code>	The capacity of the <code>books</code> array (50). Constant
<code>int: sizeUser</code>	The capacity of the <code>users</code> array (100). Constant
<code>Book array: books</code>	An array of <code>Book</code> objects
<code>User array: users</code>	An array of <code>User</code> objects
<code>int: numBooks</code>	Number of books in the database (library)
<code>int: numUsers</code>	Number of users in the database (library)
Member functions (public):	
Default constructor	Sets both <code>numBooks</code> and <code>numUsers</code> to value 0.
<code>getSizeBook()</code>	Returns <code>sizeBook</code> as an integer
<code>getSizeUser()</code>	Returns <code>sizeUser</code> as an integer
<code>getNumBooks()</code>	Returns <code>numBooks</code> as an integer
<code>getNumUsers()</code>	Returns <code>numUsers</code> as an integer
<code>readBooks(string)</code>	Takes a string (the name of the file to be read) and populates the <code>books</code> array. Returns the total number of books in <code>books</code> array as an integer
<code>readRatings(string)</code>	Takes a string (the name of the file to be read) and populates the <code>users</code> array. Returns the total number of users in <code>users</code> array as an integer
<code>printAllBooks()</code>	Prints all books stored in <code>books</code> array.

<code>getCountReadBooks(string)</code>	Takes a string (username) and returns the number of books read by that user as an integer.
<code>calcAvgRating(string)</code>	Takes a string (the title of a book) and returns the average rating of the specified book as a double
<code>addUser(string)</code>	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.
<code>checkOutBook(string, string, int)</code>	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.
<code>viewRatings(string)</code>	Takes a string (username) and prints all the books a user has provided ratings for.
<code>getRating(string, string)</code>	Takes two strings (username and book title) and returns that user's rating for the specified book.
<code>getRecommendations(string)</code>	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 2 Coderunner problems on Moodle for testing your implementations.

## Problem 1 - the member function `readBooks`

Update the `readBooks` function from Homework 7 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 input argument:
  - `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 the file is not opened successfully, return -1.
  - When `numBooks` is equal to the `size`, return -2.
  - 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 from 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

<b>fileName.txt</b>	Author A,Book 1 Author B,Book 2
<b>Function calls</b>	<pre>// make library object Library myLibrary // call readBooks int rv = myLibrary.readBooks("fileName.txt"); // print values cout &lt;&lt; "rv = " &lt;&lt; rv &lt;&lt; endl; cout &lt;&lt; "numBooks = "; cout &lt;&lt; myLibrary.getNumBooks() &lt;&lt; endl;  // print books myLibrary.printAllBooks();</pre>
<b>Output</b>	<pre>rv = 2 numBooks = 2 Here is a list of books Book 1 by Author A Book 2 by Author B</pre>

*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.

<b>book1.txt</b>	Author A,Book 1 Author B,Book 2
<b>book2.txt</b>	Author C,Book 3 Author D,Book 4
<b>Function calls</b>	<pre>// make library object Library myLibrary  // call readBooks and check return values int rv1 = myLibrary.readBooks("book1.txt"); cout &lt;&lt; "rv1 = " &lt;&lt; rv &lt;&lt; endl; int rv2 = myLibrary.readBooks("book2.txt"); cout &lt;&lt; "rv2 = " &lt;&lt; rv &lt;&lt; endl;  // check value of getNumBooks cout &lt;&lt; "numBooks = "; cout &lt;&lt; myLibrary.getNumBooks() &lt;&lt; endl;</pre>



	<pre>// print books myLibrary.printAllBooks();</pre>
<b>Output</b>	<pre>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</pre>

*Example 3:* file does not exist.

<b>Function call</b>	<pre>// make library object Library myLibrary  // call readBooks and check return values int rv1 = myLibrary.readBooks("badFile.txt"); cout &lt;&lt; "rv1 = " &lt;&lt; rv &lt;&lt; endl;</pre>
<b>Output</b>	<pre>rv1 = -1</pre>

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

<b>book1.txt</b>	<pre>Author A,Book 1 Author B,Book 2 Author C,Book 3</pre>
<b>Function call</b>	<pre>// make library obj Library myLibrary  // multiple files were read  // check value of getNumBooks cout &lt;&lt; "numBooks = "; cout &lt;&lt; myLibrary.getNumBooks() &lt;&lt; endl;  // call readBooks and check return values int rv1 = myLibrary.readBooks("book1.txt"); cout &lt;&lt; "rv1 = " &lt;&lt; rv &lt;&lt; endl;  // check value of getNumBooks</pre>

	<pre> cout &lt;&lt; "numBooks = "; cout &lt;&lt; myLibrary.getNumBooks() &lt;&lt; endl;  // print books myLibrary.printAllBooks(); </pre>
<b>Output</b>	<pre> 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 </pre>

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

<b>fileName.txt</b>	<pre> Author A,Book 1 Author B,Book 2 Author C,Book 3 </pre>
<b>Function call</b>	<pre> // make library obj Library myLibrary  // multiple files were read  // check value of getNumBooks cout &lt;&lt; "numBooks = "; cout &lt;&lt; myLibrary.getNumBooks() &lt;&lt; endl;  // call readBooks and check return values int rv1 = myLibrary.readBooks("book1.txt"); cout &lt;&lt; "rv1 = " &lt;&lt; rv &lt;&lt; endl;  // check value of getNumBooks cout &lt;&lt; "numBooks = "; cout &lt;&lt; myLibrary.getNumBooks() &lt;&lt; endl; </pre>
<b>Output</b>	<pre> numBooks = 50 rv1 = -2 numBooks = 50 </pre>

## Problem 2 - the member function `printAllBooks`

The `printAllBooks` function from Homework 7 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 arguments.
- This function does **not** return anything
- 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.

<b>Expected output</b> (assuming you have read the data from <code>books.txt</code> )
---

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 `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  
rosum,5,5,5  
...
```

This function should:

- Accept one input argument:
  - `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 Problem 3 in Homework 6, 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:
  - Return the total number of users in the system, as an integer.
  - If the file cannot be opened, return -1
  - When `numUsers` is greater than or equal to the `sizeUser`, return -2
  - The priority of the return code -2 is higher than -1, i.e., in cases when `numUsers` is equal to the `sizeUser` and the file cannot be opened, the function should return -2
  - 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

<b>bookFile.txt</b>	AuthorA,Book1 AuthorB,Book2 AuthorC,Book3 AuthorD,Book4 AuthorF,Book5
<b>ratingFile.txt</b>	Ninja,0,1,2,3,4 Myth,2,2,4,5,1 Sphyer,3,1,0,0,5 Daequan,0,0,0,0,2
<b>Function call</b>	<pre>// 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 &lt;&lt; "rv1 = "; cout &lt;&lt; rv1 &lt;&lt; endl;  // check value of getNumUsers cout &lt;&lt; "numUsers = "; cout &lt;&lt; lib.getNumUsers() &lt;&lt; endl;  // print user's ratings string name = "Ninja" cout &lt;&lt; lib.getRating(name, "book1") &lt;&lt; endl; cout &lt;&lt; lib.getRating(name, "book2") &lt;&lt; endl; cout &lt;&lt; lib.getRating(name, "book3") &lt;&lt; endl; cout &lt;&lt; lib.getRating(name, "book4") &lt;&lt; endl; cout &lt;&lt; lib.getRating(name, "book5") &lt;&lt; endl;</pre>
<b>Output</b>	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.

<b>ratingFile.txt</b>	Ninja,0,1,2,3,4 Myth,2,2,4,5,1 Sphyer,3,1,0,0,5 Daequan,0,0,0,0,2
<b>ratingFile2.txt</b>	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
<b>Function calls</b>	<pre>// make library obj Library lib  // call readRatings and check return values int rv1 = lib.readRatings("ratingFile.txt"); cout &lt;&lt; "rv1 = " &lt;&lt; rv &lt;&lt; endl; int rv2 = lib.readRatings("ratingFile2.txt"); cout &lt;&lt; "rv2 = " &lt;&lt; rv &lt;&lt; endl;  // check value of getNumBooks cout &lt;&lt; "numUsers = "; cout &lt;&lt; lib.getNumUsers() &lt;&lt; endl;</pre>
<b>Output</b>	rv1 = 4 rv2 = 9 numUsers = 9

*Example 3:* file does not exist.

<b>Function call</b>	<pre>// make library obj Library myLibrary</pre>
----------------------	--

	<pre>// call readBooks and check return values int rv1 = myLibrary.readRatings("badFile.txt"); cout &lt;&lt; "rv1 = " &lt;&lt; rv &lt;&lt; endl;</pre>
<b>Output</b>	rv1 = -1

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

<b>ratingFile.txt</b>	<pre>Ninja,0,1,2,3,4 Myth,2,2,4,5,1 Sphyer,3,1,0,0,5 Daequan,0,0,0,0,2</pre>
<b>Function call</b>	<pre>// make library obj Library myLibrary  // multiple files were read  // check value of getNumUsers cout &lt;&lt; "numUsers = "; cout &lt;&lt; myLibrary.getNumUsers() &lt;&lt; endl;  // call readRatings and check return values int rv1 = myLibrary.readRatings("ratingFile.txt"); cout &lt;&lt; "rv1 = " &lt;&lt; rv1 &lt;&lt; endl;  // check value of getNumUsers cout &lt;&lt; "numUsers = "; cout &lt;&lt; myLibrary.getNumUsers() &lt;&lt; endl;  // call readRatings again int rv2 = myLibrary.readRatings("ratingFile.txt"); cout &lt;&lt; "rv2 = " &lt;&lt; rv2 &lt;&lt; endl;</pre>
<b>Output</b>	<pre>numUsers = 98 rv1 = 100 numUsers = 100 rv2 = -2</pre>

## Problem 4 - the member function `getRating`

The member function `getRating` accepts the given a 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 input arguments 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

<b>bookFile.txt</b>	Author1,Title1 Author2,Title2 Author3,Title3
<b>ratingFile.txt</b>	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

<b>Function call</b>	<code>getRating("User1", "Title2");</code>
<b>Return value</b>	4

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



<b>Function call</b>	<code>getRating("User4", "Title1");</code>
<b>Return value</b>	-3

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

<b>Function call</b>	<code>getRating("User1", "Title10");</code>
<b>Return value</b>	-3

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

<b>Function call</b>	<code>getRating("User12", "Title10");</code>
<b>Return value</b>	-3

## Problem 5 - 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 argument:
  - `string: username`
- 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

*Example 1:* The library is initialized

<b>bookFile.txt</b>	<code>Author1,Title1</code> <code>Author2,Title2</code> <code>Author3,Title3</code>
<b>ratingFile.txt</b>	<code>User1,1,4,2</code> <code>User2,0,5,3</code> <code>User3,0,0,0</code>
<b>Function calls</b>	<code>//Create a new Library</code> <code>Library myLibrary;</code>

	<pre>//add books to Library myLibrary.readBooks("bookFile.txt");  //add users myLibrary.readRatings("ratingFile.txt");  // viewRatings for User2 cout &lt;&lt; myLibrary.getCountReadBooks("User2");</pre>
outputs	2

*Example 2: The user does not exist*

<b>bookFile.txt</b>	<pre>Author1,Title1 Author2,Title2 Author3,Title3</pre>
<b>ratingFile.txt</b>	<pre>User1,1,4,2 User2,0,5,3 User3,0,0,0</pre>
Function calls	<pre>//Create a new Library Library myLibrary;  //add books to Library myLibrary.readBooks("bookFile.txt");  //add users myLibrary.readRatings("ratingFile.txt");  cout &lt;&lt; myLibrary.getCountReadBooks("User4");</pre>
outputs	-3

*Example 3: The user has not rated any book yet*

<b>bookFile.txt</b>	<pre>Author1,Title1 Author2,Title2 Author3,Title3</pre>
<b>ratingFile.txt</b>	<pre>User1,1,4,2 User2,0,5,3</pre>

	User3,0,0,0
Function calls	<pre>//Create a new Library Library myLibrary;  //add books to Library myLibrary.readBooks("bookFile.txt");  //add users myLibrary.readRatings("ratingFile.txt");  // getCountReadBooks for User3 cout &lt;&lt; myLibrary.getCountReadBooks("User3");</pre>
outputs	0

## Problem 6 - the member function `viewRatings`

Create a new member function `viewRatings` prints all the books a user has provided ratings for. 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 input argument:
  - string: `username`
- 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, print:

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

- If the user exists in the database, and has rated at least one book, 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

<b>bookFile.txt</b>	Author1,Title1 Author2,Title2 Author3,Title3
<b>ratingFile.txt</b>	User1,1,4,2 User2,0,5,3 User3,0,0,0
<b>Function calls</b>	<pre>//Create a new Library Library myLibrary;  //add books to Library myLibrary.readBooks("bookFile.txt");  //add users myLibrary.readRatings("ratingFile.txt");  myLibrary.viewRatings("User2");</pre>
<b>outputs</b>	Here are the books that User2 rated Title : Title2 Rating : 5 ----- Title : Title3 Rating : 3 -----

*Example 2:* The user has not rated any book yet

<b>bookFile.txt</b>	Author1,Title1 Author2,Title2 Author3,Title3
---------------------	--

<b>ratingFile.txt</b>	User1,1,4,2 User2,0,5,3 User3,0,0,0
<b>Function calls</b>	//Create a new Library Library myLibrary;  //add books to Library myLibrary.readBooks("bookFile.txt");  //add users myLibrary.readRatings("ratingFile.txt");  myLibrary.viewRatings("User3");
<b>outputs</b>	User3 has not rated any books yet.

*Example 3:* The user does not exist

<b>bookFile.txt</b>	Author1,Title1 Author2,Title2 Author3,Title3
<b>ratingFile.txt</b>	User1,1,4,2 User2,0,5,3 User3,0,0,0
<b>Function calls</b>	//Create a new Library Library myLibrary;  //add books to Library myLibrary.readBooks("bookFile.txt");  //add users myLibrary.readRatings("ratingFile.txt");  myLibrary.viewRatings("User4");
<b>outputs</b>	User4 does not exist.

## Problem 7 - the member function `calcAvgRating`

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

- Accept one argument:
  - `string`: book title
- The book title search should be case insensitive. For example, “Ben”, “ben” and “BEN” are one and the same user.
- The average rating is calculated by the sum of non-zero rating values divided by the number of non-zero ratings.
- 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 title is not found
  - 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.

<b>bookFile.txt</b>	Author1,Title1 Author2,Title2 Author3,Title3
<b>ratingFile.txt</b>	User1,1,4,2 User2,0,5,3 User3,0,0,0
Function calls	<pre>//Create a new Library Library myLibrary;  //add books to Library myLibrary.readBooks("bookFile.txt");  //add users myLibrary.readRatings("ratingFile.txt");  // calcAvgRating for Title2 cout &lt;&lt; myLibrary.calcAvgRating("title2");</pre>

outputs	4.5
---------	-----

The function returns 4.5 because User 1 rated 4 and User 2 rated 5, which means  $(4 + 5) / 2 = 4.5$ . Since User 3's rating is 0, it is not included for the calculation.

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

<b>bookFile.txt</b>	Author1,Title1 Author2,Title2 Author3,Title3
<b>ratingFile.txt</b>	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");
outputs	-3 0

## Problem 8 - the member function `addUser`

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

- Accept one argument:
  - `string: user name`
- The user name is case insensitive (e.g. Ben, BEN, ben are all same as ben)
- 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: Successfully user is added to the `users` array.*

<b>ratingFile.txt</b>	User1,1,4,2 User2,0,5,3 User3,0,0,0
<b>Function calls</b>	<pre>//Create a new Library Library myLibrary;  myLibrary.readRatings("ratingFile.txt");  // checking the user count cout &lt;&lt; "numUsers = " &lt;&lt; myLibrary.getNumUsers() &lt;&lt; endl;  //add users cout &lt;&lt; myLibrary.addUser("User4");  // checking the user count cout &lt;&lt; "numUsers = " &lt;&lt; myLibrary.getNumUsers() &lt;&lt; endl;</pre>
<b>outputs</b>	<pre>numUsers = 3 1 numUsers = 4</pre>

*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)*

<b>ratingFile.txt</b>	User1,1,4,2 User2,0,5,3 User3,0,0,0
<b>Function calls</b>	<pre>//Create a new Library Library myLibrary;</pre>



	<pre>//add users myLibrary.readRatings("ratingFile.txt");  //add users cout &lt;&lt; myLibrary.addUser("user2");</pre>
outputs	0

**Example 3:** The `users` array is full

Function calls	<pre>//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 &lt;&lt; "numUsers = "; cout &lt;&lt; myLibrary.getNumUsers() &lt;&lt; endl;  // add users cout &lt;&lt; myLibrary.addUser("user4");</pre>
outputs	<pre>numUsers = 100 -2</pre>

## Problem 9 - the member function `checkOutBook`

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

- Accept three arguments 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" are one and the same user.
- The function returns the following integer value depending the cases (with the following the 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

<b>bookFile.txt</b>	<pre> Author1,Title1 Author2,Title2 Author3,Title3 </pre>
<b>ratingFile.txt</b>	<pre> User1,1,4,2 User2,0,5,3 User3,0,0,0 </pre>
<b>Set up</b>	<pre> //Create a new Library Library myLib;  myLib.readBooks("bookFile.txt"); myLib.readRatings("ratingFile.txt"); </pre>

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

Set up	<pre>int oldRating = myLib.getRating("User2", "Title1");  int rv = myLib.checkOutBook("User2", "Title1", 2);  int newRating = myLib.getRating("User2", "Title1");  cout &lt;&lt; "rv = " &lt;&lt; rv &lt;&lt; endl; cout &lt;&lt; "oldRating = " &lt;&lt; oldRating &lt;&lt; endl; cout &lt;&lt; "newRating = " &lt;&lt; newRating &lt;&lt; endl;</pre>
outputs	<pre>rv = 1 oldRating = 0 newRating = 2</pre>

*Example 2:* The rating value is invalid

Set up	<pre>int oldRating = myLib.getRating("User2", "Title1");  int rv = myLib.checkOutBook("User2", "Title1", 10);  int newRating = myLib.getRating("User2", "Title1");  cout &lt;&lt; "rv = " &lt;&lt; rv &lt;&lt; endl; cout &lt;&lt; "oldRating = " &lt;&lt; oldRating &lt;&lt; endl; cout &lt;&lt; "newRating = " &lt;&lt; newRating &lt;&lt; endl;</pre>
outputs	<pre>rv = -4 oldRating = 0 newRating = 0</pre>

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

*Example 3:* title is not found

Function calls	<pre>int rv = myLib.checkOutBook("User2", "noTitle", 1);  cout &lt;&lt; "rv = " &lt;&lt; rv &lt;&lt; endl;</pre>
outputs	<pre>rv = -3</pre>

*Example 4:* user is not found

Function calls	<pre>int rv = myLib.checkOutBook("noUser", "title1", 2); cout &lt;&lt; "rv = " &lt;&lt; rv &lt;&lt; endl;</pre>
outputs	<pre>rv = -3</pre>

## Problem 10 - the member function `getRecommendations`

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

- Accept one input argument:
  - `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" are one and 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, print the following:

```
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 the other 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.**

*john's ratings* : [5, 0, 5, 3]

*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:*

<b>bookFile.txt</b>	<code>Author1,Title1 Author2,Title2 Author3,Title3 Author4,Title4 Author5,Title5</code>
<b>ratingFile.txt</b>	<code>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</code>
<b>Set-up</b>	<code>//Create a new Library Library myLib;  myLib.readBooks("bookFile.txt"); myLib.readRatings("ratingFile.txt");</code>

*Example 1:* There are books to recommend

Function call	<code>myLib.getRecommendations("User5");</code>
outputs	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: No books to recommend*

Function call	<code>myLib.getRecommendations("User2");</code>
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. Hence, no books to recommend.

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

Function call	<code>myLib.getRecommendations("User4");</code>
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

(30 points in codeRunner)

### Problem 11 - Driver function

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**, to modify for this problem so we can show off to our enemies how much 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 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.
  - 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.
- 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: Get a rating**
  - If the database has not been initialized, print
    - `Database has not been fully initialized`
  - Otherwise:
    - Prompt the user for a username.
    - Prompt the user for a 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 5:** 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.
    - 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 6:** 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.
    - Pass the username to your `viewRatings` function
- **Option 7:** 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.
    - 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 8:** Add a user to the database.

- Prompt the user for a username.
- 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 9:** 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.
  - Prompt the user for a title.
  - Prompt the user for 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 10:** get recommendations

- If the database has not been initialized, print

- Database has not been fully initialized
- Otherwise:
  - Prompt the user for a username.
  - Pass the username to your `getRecommendations` function
- **Option 11: Quit**
  - Print "good bye!" before exiting

## 5. 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 \(File Submission\)](#). The zip file should be named, **<firstName>\_<lastName>\_project2.zip**., and have following 7 files:

1. Book.h
2. Book.cpp
3. User.h
4. User.cpp
5. Library.h
6. Library.cpp
7. project2.cpp

3. Sign up to the interview grading slot on Moodle. Please make sure that you sign-up and complete an interview grading with your TA by **Monday, April 15**. The schedulers for interview grading will be available after the deadline of this project.

You are allowed to reschedule an interview grading session without any penalty once during the semester. If you miss a second time, you may reschedule but there will be a 25-point (out of 100) penalty, then 50 points for the third time.

## 6. Project 2 point summary

Criteria	Pts
Coderunner	180
Interview grading	60
Style and Comments	5
Algorithms	5
Recitation attendance (Mar 19 or Mar 21)*	-50
Not using Library class in the driver	-30
Using global variables	-5
Total	250

\* If your attendance is not recorded, you will lose points.

Make sure your attendance is recorded on Moodle **before you leave recitation**.