CSCI 1300 CS1: Starting Computing
Ashraf, Cox, Spring 2020
Project 2: Recommender system
Due: Wednesday, April 1, by 11:50 pm

Early submission bonus:

- 5% bonus by submitting all parts by Friday March 20
- 3% bonus by submitting all parts by Sunday March 29
- 2% bonus by submitting all parts by Tuesday, March 31

Objectives

- Be able to use topics learned this semester (control structures, array, fileIO)
- Understand classes and create objects
- Develop a menu driven program

Submissions

- **Project 2 survey**. There are a few survey questions about this project. Don't forget to respond to them!

- **h files and C++ files**. All files should be named as specified in each question, and they should compile and run on Cloud 9 to earn full points. TAs will be grading styles of your code and comments. Please see the style guide on Moodle. At the top of each file, write your name with the following format:

  ```
  // CS1300 Spring 2020
  // Author: Punith Sandhu
  // Recitation: 123 - Favorite TA
  // Project2 - Problem # …
  ```

  For each class, you should have a h file and cpp file nicely organized. As you implement each method in the classes, you need to test and make sure that it works. **Here is an example**.

- **Code runner**. Your program will be graded by the code runner. You can modify your code and re-submit (press Check again) as many times as you need to, up until the assignment due date.

- **Interview grading**. Sign up, between Mar. 18 and Mar. 31, for the interview grading slot on Moodle. If you don't sign-up between Mar. 18 and Mar. 31 and you miss your interview, then no points will be awarded for the project.

Tips!

1) Your code should always compile and run.
2) Be sure to test each method as you write it. You should have your own main for testing.
3) This project looks long, but it's not.
4) Work a little bit every single day, rather than all at once
5) Create helper functions. Other than the required questions, you can have your own helper methods. That would be helpful and makes it easier for you.

Background

If you've ever bought a book online, the bookseller's website probably told you what other books you might like. This is handy for customers, but also very important for business.

In 2009, online movie-rental company Netflix awarded one million dollars to the winners of the Netflix Prize. The competition simply asked for an algorithm that would perform 10% better than their own algorithm. Making good predictions about people's preferences was that important to this company. It is also a very happening field of research that lies at an intersection between math, machine learning and computer science.

So how might we write a program to make recommendations for books?

Consider a user named Rabia. How is it that the program should predict books Rabia might like? The simplest approach would be to make almost the same prediction for every customer. In this case, the program would simply calculate the average rating for all the books in the database, sort the books by rating and then from that sorted list, suggest the top 5 books that Rabia hasn't already rated. With this simple approach, the only information unique to Rabia used by the prediction algorithm was whether or not Rabia has read a specific book.

We could make a *better* prediction about what Rabia might like by considering her actual ratings in the past and how these ratings compare to the ratings given by other customers. Consider how you decide on movie recommendations from friends. If a friend tells you about a few movies that (s)he enjoyed and you also enjoyed them, then when your friend recommends another movie that you have never seen, you probably are willing to go see it. On the other hand, if you and a different friend always tend to disagree about movies, you are not likely to go to a movie this friend recommends.

A program can calculate how similar two users are by treating each of their ratings as an array/vector and calculating a similarity value based on some calculation (like a dot product, or a sum of squared differences) using the two arrays.

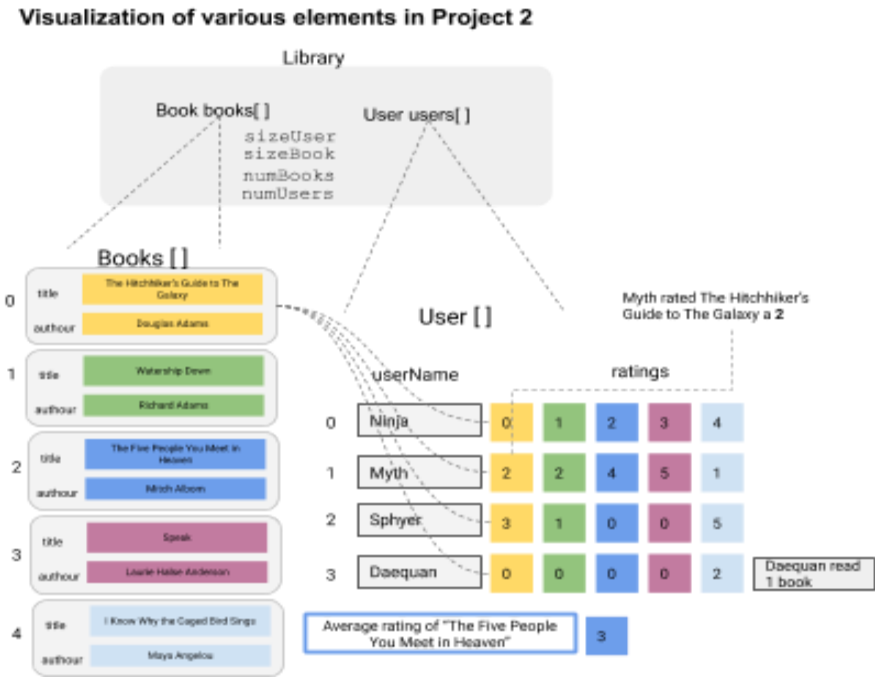
Once you have calculated the pairwise similarity between Rabia and every other customer, you can then identify whose ratings are most similar to Rabia's. If another user, Suelyn, is most similar to Rabia, we would recommend to Rabia the top books from Suelyn's list that Rabia hasn't already rated.

Project Goal and Design

In project 2, you will be creating a book recommender system. You will be creating a **Library** class that comprises objects from **Book** and **User** classes.

Specifications

- Create a new class **Book,** where each instance represents a book and the **Library** class stores an array of these books. Define the class in a header file and implement it in a separate cpp file.
- Create a new class **User,** where each instance represents a person and the ratings they have given to books.The **Library** class stores an array of these users. Define the class in a header file and implement it in a separate cpp file.
- Create a new class **Library**. Define the class in a header file and implement it in a separate cpp file.
- You will have to create an instance of **Library** class in **main()** and call the respective functions to perform various tasks.



Tasks: Let's do this!

Task 1(10pt): Book class

Create a `Book` class, with a separate interface file (`Book.h`) and an implementation file (`Book.cpp`), comprised of the following attributes:

The `Book` class has the following attributes:

Data members (private):	
`title`: `string`	The title of the book
`author`: `string`	The author of the book
Member functions (public):	
Default constructor	Set `title` and `author` to an empty `string`
Parameterized constructor	Takes two `strings` assigning `title` and `author`, in this order
`getTitle()`	Returns the `title` as a `string`
`getAuthor()`	Returns the `author` as a `string`
`setTitle(string)`	Sets the player's `name` (and returns nothing)
`setAuthor(double)`	Sets the player's `points` (and returns nothing)

Sample test cases (Be sure to test all methods!)	Expected outputs

`Book hungerGame("The Hunger Games", "Suzanne Collins");`
`cout << hungerGame.getTitle() << endl;`
`cout << hungerGame.getAuthor() << endl;`
	`The Hunger Games`
	`Suzanne Collins`

For the zip submissions, the files should be named as `Book.h` and `Book.cpp`. Your implementation should be organized nicely into separate files. For the code runner, paste your `Book` class and its implementation (both `Book.h` and `Book.cpp`) in the answer box. Please make sure to test your `Book` class on your Cloud 9 / VS code before submitting it to the code runner.

Task 2(15pt): User class

Create a `User` class, with a separate interface file (`User.h`) and an implementation file (`User.cpp`), comprised of the following attributes:

The `User` class has the following attributes:

Data members (private):	
`username: string`	The name of the user
`ratings: an array of integer`	An array of integers (size 50) where all the ratings are stored
Member functions (public):	
Default constructor	Set `username` to an empty `string`, and initialize all elements of the `ratings` array to `0`
`setUsername(string)`	Takes a `string` to set `username` (and returns nothing)
`setRatingAt(int, int)`	Takes an index `i` and a rating value. If the index and the rating value are valid, assign the rating at the index.
`getUsername()`	Returns the name of this user (`string`)
`getRatingAt(int)`	If `i` is within the bounds of the `ratings` array, it returns the rating value at index `i` of the ratings array. If not, it returns -1.

`User` class method: `setRatingAt(int, int)`

This setter takes the index of the rating to be set, and the rating value to set it to. If the index is within the bounds of the `ratings` array and the rating value is between 0 and 5, the function sets the rating value at the index and returns true. Otherwise, it returns false.

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!

Method specifications:

- The method name: **`setRatingAt`**
- The method takes parameters in this order:
    - An index of the book, `int`
    - A new rating value, `int`
- The method sets the new rating value if the index is within the bounds of the rating array and the rating value is valid (i.e. between 0 and 5).
- The method returns a boolean value depending on the following cases:
    - True if it successfully updated the rating
    - False if it did not

`User` Class Method: `getRatingAt(int)`

This getter takes the index of a rating. If the index is within the bounds of the `ratings` array, then it returns the rating value at the index. If not, it returns -1.

Method specifications:

- The method name: **`getRatingAt`**
- The method takes parameters in this order:
    - An index of the book, `int`
- The method returns an integer value:
    - It returns the rating value if the index is within the boundary of the rating array
    - It returns -1 if the index is outside of the bounds of the ratings array

Sample test cases (Be sure to test all methods!)	Expected outputs

`User hector;`
`hector.setUsername("Hector Ramirez");`
`hector.setRatingAt(0,4);`
`cout << hector.getUsername() << endl;`
`cout << hector.getRatingAt(0) << endl;`
	`Hector Ramirez`
	`4`

For the zip submissions, the files should be named as `User.h` and `User.cpp`. For the code runner, paste your `Book` class and its implementation (both `User.h` and `User.cpp`) in the answer box. Please make sure to test your `User` class on your Cloud 9 / VS code before submitting it to the code runner.

Task 3(80pt): Library Class

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

The `Library` class has the following attributes:

Data members (private):	
`books`: an array of `Book` objects	The books in the library. Its size is 50 (constant)
`users`: an array of `User` objects	The users in the library. Its size is 100 (constant)
`numBooks`: `int`	The number of books in the library
`numUsers`: `int`	The number of users in the library
Member functions (public):	
Default constructor	Sets both `numBooks` and `numUsers` to 0
`getNumBooks()`	Returns `numBooks` as an `integer`
`getNumUsers()`	Returns `numUsers` as an `integer`
`readBooks(string)`	Takes a file name (a `string`), reads a list of books, and stores them into the `books` array for this library. It returns the total number of books stored in the `books` array as an `integer`.
`readRatings(string)`	Takes a file name (a `string`), reads a list of users and their ratings and stores them into the `users` array for this library. It returns the total number of users stored in the `users` array as an `integer`.
`viewRatings(string, int)`	Takes a username and a minimum rating value. It prints all the books that the user has rated with a value greater than or equal to the minimum rating value. The function does not return anything.
`printAllBooks()`	Prints all books stored in the `books` array with average ratings. The function does not return anything.
`addUser(string)`	Takes a username and adds a user with that name to the `users` array. The function does not return anything.
`updateRating(string,string`	Takes username, title, and a new rating (in this order)

`,int)`	and updates the rating value of this title for this user. The function does not return anything.
`getRecommendations(string)`	Takes the username and prints the first 5 book recommendations. The function does not return anything.

`Library` Class Method: **`readBooks(string)`**

It takes a file name, reads a list of books, and stores them into the `books` array. Each line has an author and a title separated by a comma. The function returns the number of the total books stored in the `books` array. If the file cannot be opened, the function should return `-1`.

Sample file (**books.txt**):

```
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
Maya Angelou,I Know Why the Caged Bird Sings
Jay Asher,Thirteen Reasons Why
```

Method specifications:

- The method name: **`readBooks`**
- The method parameter:
  - A filename, `string`
- The function returns an integer value depending on the following conditions (checked in this order)
  - It returns `-1` if the file cannot be opened.
  - It returns the total number of books if the `books` array is full
  - It returns the total number of books stored in the array if it can successfully read the list of books.
- Important:
  - The method stores all the books stored without replacing them. For example, if it reads a file with 5 books (the first time it is called) and then another file with 3 books (the second time it is called), then all 8 books should be stored in the `books` array and it returns 8 (the second time it is called).

How to test it?

- You can test the method is working or not by 1) calling it in your main and see if it returns the correct value, 2) calling it multiple times to see if the function returns the total number of the books stored in the array, 3) implement the other methods (`viewRatings` and `printAllBooks`) to see if it prints the books stored in the array

`Library` Class Method: **`readRatings(string)`**

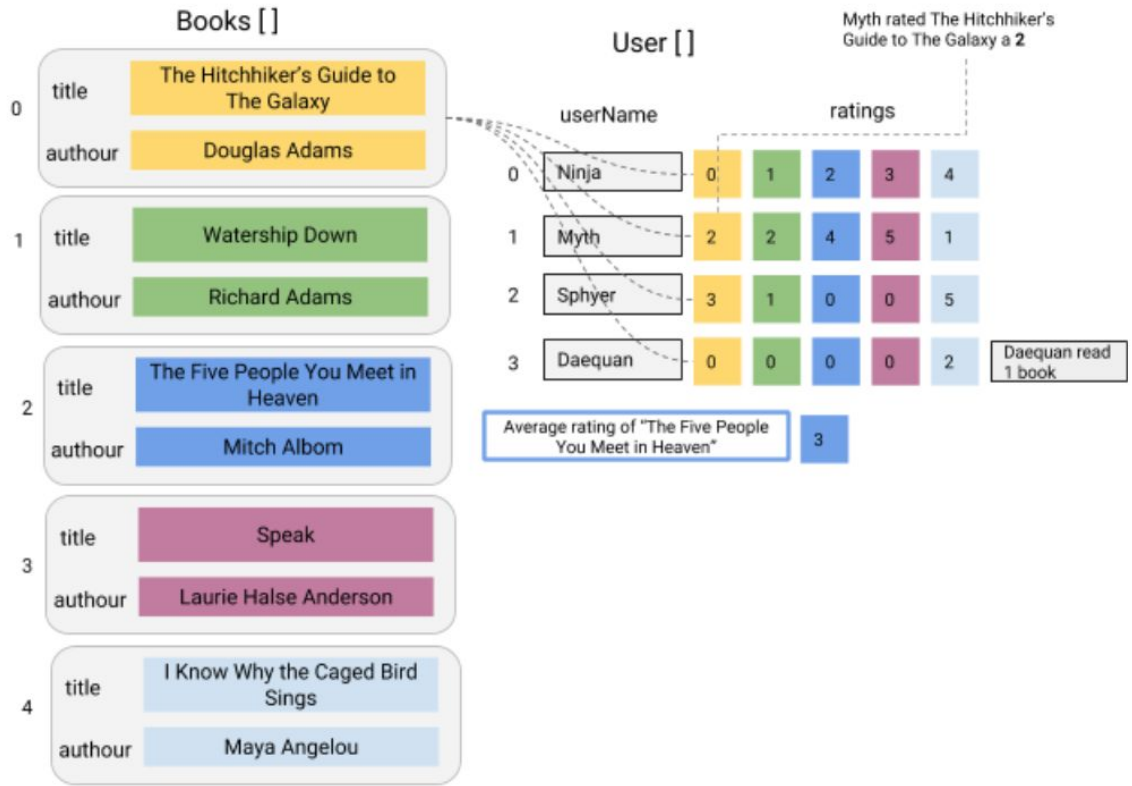
It takes a file name and reads usernames and their ratings. A username is stored on one line, and the user's ratings are stored on the following line, separated by commas.

Sample file (**ratings.txt**):

```
cynthia
4,3,1,0,3,0,5,1,5,2,2,2,1,4,4,2,0,1,1,2
diane
3,1,1,0,2,2,3,1,0,1,4,3,1,2,1,1,5,2,4,0
joan
3,1,2,0,2,2,4,3,0,2,2,4,5,2,2,1,4,1,1,3
```

In the above sample file, Cynthia rated 4 for the book at index 0 , rated 3 for the book at index 1, and so on. Diane rated 3 for the book at index 0, rated 1 for the book at index 1.

Each rating is mapped to the book by the index. As shown in below, the rating value at index 0 of the `ratings` array (for any particular user) will be that user's rating for the book at index 0 of the `books` array. The same holds for ratings and books at all other indices of the `ratings` and `books` arrays.



Method specifications:

- The method name: **`readRatings`**
- The method parameter:
    - A filename, `string`
- The function returns an integer value depending on the following conditions (in this order)
    - It returns -1 if the file cannot be opened.
    - It returns the total number of users if the `users` array is full
    - It returns the total number of users stored in the array if it can successfully read the list of users and ratings.
- Important:
    - The method stores all the users stored without replacing them. For example, if it reads a file with 5 users (the first time it is called) and then another file with 3 users (the second time it is called), then all 8 users and their ratings should be stored in the `users` array and it returns 8 (the second time it is called).

How to test it?

You can test the method is working or not by 1) calling it in your main and see if it returns the correct value, 2) calling it multiple times to see if the function returns the total number of the users stored in the array, 3) implement the other methods (`viewRatings` and `printAllBooks`) to see if it prints the users and their ratings stored in the array

`Library` Class Method: `viewRatings(string, int)`

It takes a username and a minimum rating value. It prints all the books the user has rated with a value that is greater than or equal to the minimum rating value in the format as shown below. The username search should be case insensitive (i.e. `Ben`, `BEN`, `BeN` are the same `ben`). Recall that a rating value of 0 means that the user has not rated the book yet.

Expected output format:

```
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
-----
```

Method specifications:

- The method name: **`viewRatings`**
- The method parameters (in this order):
    - A username, `string`

    - A minimum rating, `int`
- The function does not return anything.

Edge cases (should be handled in this order):

- The library has not been initialized yet. If there are no books or no users in the array, it cannot print the ratings. Display the following message:

  ```
  The library has not been fully initialized.
  ```

- There might be a case where the user does not exist. When you search for a username, it should be case insensitive. If the username is not found, then it displays the following message. `<username>` is an actual username passed into the method.

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

- If you find a username, but there are no ratings that are greater than or equal to the minimum rating value, display the following message:

  ```
  <username> has not rated any books with <min rating> or higher.
  ```

Sample Run 1 for **Akriti** with minRating = 4 (**books_mini.txt** and **users_mini.txt**):

```
Here are the books that Akriti rated
Title : Title1
Rating : 5
-----
Title : Title5
Rating : 4
-----
```

Sample Run 2 for **Keya** with minRating = 2 (**books_mini.txt** and **users_mini.txt**):

```
Keya has not rated any books with 2 or higher.
```

How to test it?
After creating an instance of the `Library` class, you can call `readBooks` and `readRatings` methods, then call `viewRatings` to see if you can get the same output as above.

`Library` Class Method: **`printAllBooks()`**

It would be helpful to see a list of all the `books` stored in the array, along with their average ratings. The average rating should be formatted with two-digit precision. If the rating value is 0, it means that the user has not rated the title yet. So we don't include it in the calculation.

Methods specifications:

- The method name: **`printAllBooks`**
- The method does not take any parameters.
- The function does not return anything.

Edge case:

- The library has not been initialized yet. If there are no books or no users in the array, it cannot print the ratings. Display the following message:

```
The library has not been fully initialized.
```

Sample Run: (**books_mini.txt** and **users_mini.txt**):

```
(3.33) Title1 by Author1
(3.00) Title2 by Author2
(2.67) Title3 by Author3
(4.00) Title4 by Author4
(3.00) Title5 by Author5
```

How to test it?
After creating an instance of the `Library` class, you can call `readBooks` and `readRatings` methods, then call `printAllBooks` to see if you can get the same output as above.

`Library` Class Method: **`updateRating(string,string,int)`**

It takes a username, title of a book, and the user's new rating for the book. It updates the rating value if the username and title exist in the arrays and the rating value is valid (between 0 and 5). When you search for a username and title, it should be case insensitive (i.e. `Ben`, `BEN`, `BeN` are the same `ben`).

Methods specifications:

- The method name: **`updateRating`**
- The method parameters (in this order):
  - A username, `string`
  - A title, `string`
  - A new rating, `int`
- The function does not return anything.

cases: (should be handled in this order):

- The library has not been initialized yet. If there are no books or no users in the array, it cannot print the ratings. Display the following message:

```
The library has not been fully initialized.
```

- There might be a case where the user does not exist. When you search for a username, it should be case insensitive. If the username is not found, then it displays the following message. `<username>` is an actual username passed to the method.

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

- If the new rating value is invalid, we cannot update the rating value. Valid ratings are between 0 and 5 (inclusive). If the new rating value is invalid, then display the following message:

  ```
  <rating> is not valid.
  ```

- There is also a case where the title does not exist in the library. If so, display the following message: `<title>` is an actual book title passed to the method.

  ```
  <title> does not exist.
  ```

- Once we check all the above conditions, we can update the user's rating for the given title. If it is successful, then display the following message:

  ```
  The rating has been updated.
  ```

Library Class Method: **addUser(string)**

We always get new users. Let's create a method to add a new user to the library. It takes a username and adds it to the `users` array, provided the username does not already exist and there is a space in the array. When you search for a username, it should be case insensitive (i.e. `Ben`, `BEN`, `BeN` are the same `ben`).

Methods specifications:

- The method name: **`addUser`**
- The method parameter:
  - A username, `string`
- The function does not return anything.
- The new user has not rated any books yet. So all the ratings for the new user should be 0.

cases: (should be handled in this order):

- The library might not have a space to add a new user. (It already has 100 users). If so, display the following message. `<username>` is an actual username passed into the method.

  ```
  The library is already full. <username> was not added.
  ```

- There might be a case where the user already exists. When you search, it should be case insensitive (i.e. Ben, ben, BEN are the same ben). If the `username` already exists in the users array, print the following message:

  ```
  <username> does already exist.
  ```

- Once you check that there is a space in the `users` array and the username does not exist, then the username should be added. If successful, print the following message:

  ```
  Welcome to the library <username>
  ```

`Library` Class Method: **`getRecommendations(string)`**

It will recommend book titles a user might enjoy based on the ratings of another user who likes similar books.

Methods specifications:

- The method name: **`getRecommendations`**
- The method parameter:
  - A username, `string`
- The function does not return anything.

How to find books to recommend?
You recommend books based on the most similar user. You can calculate how similar two users are by calculating the sum of square differences (SSD).

The SSD is calculated by summing the squares of the differences between the corresponding elements in the two ratings array of two users. Because our similarity metric is based on difference, more similar users will have smaller similarity values. Let's say we want to generate recommendations for Ben. In the equation below, `A` represents Ben's rating for `i`th book and B represents the other user's rating for `i`th book.

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

Once you find the user most similar to the given user (Ben in this example), print the first 5 books that the most similar user rated as 3 or higher, that the given user (Ben) has not rated yet (i.e. the rating is 0).

Note: If the user has not rated any books, then the user cannot be the most similar user.

For example, suppose we have 5 books (stored in the books array in this order):

Lab Girl
The Rosie Project
The Catcher in the Rye
Unbelievable
A Man of the People

And 5 users in the users array in the library:

Akriti:	[5, 0, 2, 0, 4]
Malvika:	[4, 3, 3, 5, 4]
Vipra:	[0, 0, 0, 0, 1]
Monika:	[0, 0, 0, 0, 0]
Keya:	[1, 0, 3, 3, 0]

Based on the above data, If you find recommendations for Akriti, then calculate the similarity between:

Akriti and Malvkia: $(5-4)^2 + (0-3)^2 + (2-3)^2 + (0-5)^2 + (4-4)^2 = 36$
Akriti and Vipra: $(5-0)^2 + (0-0)^2 + (2-0)^2 + (0-0)^2 + (4-1)^2 = 38$
Akriti and Monika: $(5-0)^2 + (0-0)^2 + (2-0)^2 + (0-0)^2 + (4-0)^2 = 45$
Akriti and Keya: $(5-1)^2 + (0-0)^2 + (2-3)^2 + (0-3)^2 + (4-0)^2 = 42$

Once you calculate the pairwise SSD between Akriti and every other user, you can then identify the user whose ratings are most similar to Akriti's. In this case, 17 is the smallest SSD value among the users. Therefore, Malvika is the most similar to Akriti.

Once you figure out the most similar user, print the first 5 books the most similar user (Malvika) rated 3 or higher, that the given user (Akriti) has not rated yet. In this case, it prints the following books:

The Rosie Project
Unbelievable

Akriti has not rated Book1 and Book3 and Malvika rated those books 3 o
r higher, so it prints them.

Based on the above data, If you find recommendations for Vipra, then we do the same calculation for the SSD score:

- Vipra and Akriti: SSD = 38
- Vipra and Malvika: SSD = 68
- Vipra and Monika: SSD = 1
- Vipra and Keya: SSD = 20

In this case, the most similar user should be Keya. The SSD between Vipra and Monika are the lowest (SSD = 1). However, since Monika has not rated any books yet (all the ratings are 0), she cannot be the most similar user.

Once you figure out the most similar user, you find the books in which Vipra has not rated and Keya rated 3 or higher. Then, it will prints:

The Catcher in the Rye
Unbelievable

Vipra has not rated the first four books, but of these, the books Keya liked (rating is 3 or higher) are only The Catcher in the Rye and Unbelievable.

Edge cases (in this order):

- The library has not been initialized yet. If there are no books or no users in the library, it cannot find recommendations. Display the following message:

  ```
  The library has not been fully initialized.
  ```

- There might be a case where the user does not exist. When you search, it should be case insensitive (i.e. Ben, ben, BEN are the same ben). If the `username` does not exist in the users array, print the following message:

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

- After you find the most similar user, but there might be no books to recommend at the moment. If so, display the following message:

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

- There might be the case where there are more than 5 books to recommend. In that case, we recommend only the first 5 books.

Sample Run: Recommendations for **Akriti**: (**books_mini.txt** and **users_mini.txt**):

```
Here is the list of recommendations
The Rosie Project by Graeme Simsion
Unbelievable by Katy Tur
```

Sample Run: Recommendations for **Vipra**: (**books_mini.txt** and **users_mini.txt**):

```
Here is the list of recommendations
The Catcher in the Rye by J. D. Salinger
Unbelievable by Katy Tur
```

Sample Run: Recommendations for **Malvika**: (**books_mini.txt** and **users_mini.txt**):

```
There are no recommendations for Malvika at present.
```

The most similar user for Malvika is Akriti (SSD = 36). However, Malvika has rated all the books Akriti has rated, so there are no books to recommend.

For the zip submissions, the files should be named as `Library.h` and `Library.cpp`. For the code runner, paste your `Book` class and its implementation, `User` class, `Library` class and its implementation (both `Book.h`, `Book.cpp`, `User.h`, `User.cpp`, `Library.h` and `Library.cpp`) in the answer box. Please make sure to test your `Library` class on your Cloud 9 / VS code before submitting it to the code runner.

Task 4(10pt): Let's make `main()`

Now, let's put it all together in the `main()` function. Create a `project.cpp` and write the `main()` function in the file. To make it user-friendly, we'll create a menu option and call the appropriate methods (that you created) for each option.

The menu has the following options:

```
Select a numerical option:
======Main Menu=====
1. Read books
2. Read ratings
3. View ratings
4. Print all books
5. Update a rating
6. Add a user
7. Get recommendations
8. Quit
```

The menu will run in a loop, continually offering the user the options until they opt to quit. You need to fill in the code for each of the options. Be sure to use the methods you wrote in Task 3.

Invalid option: If the user enters invalid option, then print

```
Invalid option.
```

Option1: Read books

The program asks for a filename of a book list and displays the total number of the books stored in the array. If the file cannot be opened, then it prints `No books saved to the database.`

Sample Run 1 (**bold** is user input)

```
Enter a book file name:
books.txt
```

```
Total books in the database: 50
```

Sample Run 2 (**bold** is user input)

```
Enter a book file name:
this_file_does_not_exist.txt
No books saved to the database.
```

Option2: Read ratings

The program asks for a filename of a rating list and displays the total number of the users stored in the array. If the file cannot be opened, then it prints `No users saved to the database.`

Sample output 1 (**bold** is user input)

```
Enter a user file name:
ratings.txt
Total users in the database: 50
```

Sample Run 2 (**bold** is user input)

```
Enter a user file name:
this_file_does_not_exist.txt
No users saved to the database.
```

Option3: View ratings

The program asks for a username and a minimum rating value. Then, it calls the **`viewRating`** method.

Sample Run (**bold** is user input)

```
Enter a user name:
Akriti
Enter a minimum rating:
2
Here are the books that Akriti rated
Title : Lab Girl
Rating : 5
-----
Title : The Catcher in the Rye
Rating : 2
-----
Title : A Man of the People
Rating : 4
-----
```

Option4: Print all books
If option 4 is selected, then it will call the **`printAllBooks`** method.

Option5: Update user's rating
The program asks for a username, a title, and a new rating. Then, it will call the **`updateRating`** method.

Sample Run (**bold** is user input)

```
Enter a user name:
Akriti
Enter a book title:
The Catcher in the Rye
Enter a new rating:
3
The rating has been updated.
```

Option6: Add a user
The program asks for a username. Then, it will call the **`addUser`** method.

Sample Run (**bold** is user input)

```
Enter a user name:
Divya
Welcome to the library Divya
```

Option7: Get recommendations
The program asks for a username. Then, it will call the **`getRecommendations`** method.

Sample Run 1 (**bold** is user input)

```
Enter a user name:
Akriti
Here is the list of recommendations
The Rosie Project by Graeme Simsion
Unbelievable by Katy Tur
```

Option8: Quit
Before exiting the program, print `Good bye!`

For the zip submissions, the files should be named as `project.cpp`. For the code runner, paste your `main()`, `Book` class and its implementation, `User` class, `Library` class and its implementation (both `Book.h`, `Book.cpp`, `User.h`, `User.cpp`, `Library.h` and `Library.cpp`) in the answer box. Please make sure to test your `main` on your Cloud 9 / VS code before submitting it to the code runner.

Extra credit

You can get extra credit points if you create new useful menu options such as add a new book, calculate average by an author, etc.. TAs will be checking it during the interview grading and add 5pt - 10pt per feature. You can earn at most 20pt by making new features.

Project 2 checklist

Here is a checklist for submitting the assignment:

1. Complete the **Project 2 survey**
2. Complete the code **Project 2 CodeRunner**
3. Submit one zip file to **project 2 zip file submission**. The zip file should be named, **project2_lastname.zip**. It should have the following 7 files:
    - **Book.h**
    - **Book.cpp**
    - **User.h**
    - **User.cpp**
    - **Libear.h**
    - **Library.cpp**
    - **project.cpp**
4. Sign up, between Mar. 18 and Mar. 31, for the interview grading slot on Moodle. The interviews will take place between Apr. 2 and Apr. 21. If you don't sign-up between Mar. 18 and Mar. 31 and you miss your interview, then no points will be awarded for the project.

Project 2 points summary

Criteria	**Pts**
Project 2 survey	5
CodeRunner (Task 1 - 4)	115
Interview grading (including comments and style)	80
Recitation attendance (Match 16, 17)*	-30
Total	200
Extra credit / Early submission bonus	30(possible)

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