**CSCI 1300 - Starting Computing**
**Instructor: Fleming**
**Homework 6: Part I**

**The work you do in this assignment will be used as part of the upcoming assignments and project 2.

**Due Sunday, October 21st, by 6 pm**
**+5% bonus if submitted by Friday October 19th 11:55 pm,**
**+2% bonus if submitted by Saturday October 20th 11:55 pm**

This assignment is due **October 21st, by 6 pm**

- ***All components*** **(Cloud9 workspace, moodle quiz attempts, and zip file) must be completed and submitted by Sunday, October 21st, by 6:00 pm for your homework to receive points.**
- Complete submissions (Cloud9 workspace, moodle quiz attempts, and zip file) before **Friday October 19th 11:55 pm** will receive a 5% bonus, and complete submissions before **Saturday October 20th 11:55 pm** will receive a 2% bonus.

---

Objectives:

- Use filestream objects to read data from files
- Array operations: initialization, search
- Improve proficiency with loops and strings

## Background

Adapted from http://nifty.stanford.edu/2011/craig-book-recommendations/cs1/handout.shtml

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 September, 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 current area of research in machine learning, which is part of the area of computer science called artificial intelligence.

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 had 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 number of 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 see a movie this friend recommends.

A program can calculate how similar two users are by treating each of their ratings as a 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 pair-wise 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 hadn't already rated.

Your task for part one is to develop logistical methods for initializing the database and manipulating its contents. In upcoming assignments you will exploit these operations for making recommendations.

## Problem Set

**New rule: **No global variables!**
**For coderunner : **Paste your complete solution including your main()**
***All the examples and values used in examples are imaginary and randomly generated.*

You were given two text files with comma separated values: `books.txt`, which is a list of books and their authors, and `ratings.txt`, which is a list of users and their ratings of those books. The first task is to read these files and load their contents into arrays for convenient processing.

| **Sample lines from `books.txt`:** |
|---|
| `Douglas Adams,The Hitchhiker's Guide To The Galaxy` |

```
Richard Adams,Watership Down
Mitch Albom,The Five People You Meet in Heaven
(etc.)
```

- *The format is <author name>,<title of the book>*

**Sample lines from `ratings.txt`:**

```
cynthia,4 3 1 0 3 0 5 1 5 2 2 2 1 4 4 2 0 1 1 2 3 2 1 1 3 4 1 2
1 3 0 0 3 1 1 3 2 3 1 2 3 4 5 5 0 1 3 2 2 4
diane,3 1 1 0 2 2 3 1 0 1 4 3 1 2 1 1 5 2 4 0 3 2 1 5 4 5 0 2 3
3 5 2 2 1 4 5 2 4 5 2 3 3 5 5 4 1 3 4 2 3
(etc.)
```

- *The format is <username>,<rating_0> <rating_1> <rating_2> … <rating_50>*
- *The order of ratings, rating_0, rating_1, rating_2,... correspond to the order of books in the `books.txt` file*

*Note: For this homework, assume there are 50 books and 100 users at the max(86 users to be exact).*

## Question 1

Write a function `readBooks` that populates a pair of arrays with the titles and authors found in `books.txt`. This function should:

- Accept five input arguments in this order:
  - string: the name of the file to be read
  - string array: titles
  - string array: authors
  - int: the number of books currently stored in the arrays
  - int: capacity of the authors/titles arrays *[assume to be 50]*
- Use `ifstream`, `stringstream`, and `getline` to read and parse data from the file, placing author names in the authors array and titles in the title array
- Return the total number of books in the system, as an integer.
- If the file cannot be opened, return -1

*Hint: You can use the split() function from recitation*
*Note: Although the user can specify the name of the file to read from, assume the default filename to be "books.txt"*

## Question 2

Write a function `readRatings` that performs a similar task on the user ratings file. Each username represented in `ratings.txt` is followed by list of integers--ratings of each book in `books.txt`.

| **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! |

Your function should:

- Accept six arguments in this order:
    - string: the name of the file to be read
    - string array: usernames
    - 2D int array: list of ratings for each user (first index specifies user)
    - int : number of users currently stored in the arrays
    - int: row capacity of the 2D array (convention: array[row][column]) *[assume to be 100]*
    - int: column capacity of the 2D array *[assume to be 50]*
- Use `ifstream`, `stringstream`, and `getline` to read and parse data from the file, placing usernames in the usernames array and book ratings in the ratings array (`stoi` will also be useful here)
- Print the username of each user as they are added to the system

```
cout << username << "..." << endl;
```

- Return the total number of users in the system, as an integer.
- If the file cannot be opened, return -1

*Hint: You can use the split() function that was used during recitation*

*Note: Although user can specify the name of the file to read from, assume the default filename to be "ratings.txt"*

| **Expected output:** |
|---|
| `cynthia...`<br>`diane...`<br>`joan...`<br>`barbara...`<br>(etc.) |

## Question 3

It will be useful to display the contents of your library. Next, make a function `printAllBooks` that meets the following criteria:

- Accept three arguments in this order:
    - string array: titles
    - string array: authors
    - int: number of books
- This function does not return anything
- If the number of books is 0, print "`No books are stored`"
- Otherwise, print "`Here is a list of books`" and then each book in a new line using the following statement

```
cout << titles[i] << " by " << authors[i] << endl;
```

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

```
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
```
(etc.)

## Question 4

Write a function `getUserReadCount` for determining how many books a particular user has read and reviewed. This function should:

- Accept five arguments in this order:
    - string: username for whom we want a read count
    - string array: all users
    - 2D int array: list of all ratings, one row for each user
    - int: number of users in the arrays
    - int: number of books accounted for in the 2D array
- Return the number of books read/reviewed by the specified user, as an integer.
- If the program has not read `ratings.txt` or `books.txt`, it must read it first before executing this function. In this case, return -1 after printing the following message:

    ```
    cout << name << " does not exist in the database" << endl;
    ```

- If instead the database is initialized but the user is not found, return -1 after printing the following message :

    ```
    cout << name << " does not exist in the database" << endl;
    ```

- Highly recommend: Write a helper function that searches the user array for a particular username and returns its index.

## Question 5

Finally, create a function `calcAvgRating` that returns the average (mean) rating for a particular book. This function should:

- Accept five arguments in this order:
    - string: book title for which you want the average rating
    - string array: titles
    - 2D int array: list of ratings for each user (same comment here)
    - int: number of users in the arrays
    - int: number of books accounted for in the 2D array
- Return the average rating of the specified book as a `double`

- If the program has not read `ratings.txt` or `books.txt`, it must read it first before executing this function. In this case, return -1 after printing the following message:

```
cout << bookTitle << " does not exist in the database" << endl;
```

- If instead the database is initialized but the book is not found, return -1 after printing the following message:

```
cout << bookTitle << " does not exist in the database" << endl;
```

- Highly recommend: Write a helper function that searches the titles array for a particular book and returns its index.

*Note: If the user has not reviewed the book it should **not** be added while calculating the average.*

## Driver function

Menu functionality has been provided for you in the starter code on Moodle in the file `HW6.cpp` . Download the file and fill in the missing parts in the code which are indicated by comments in the `main()` function.

**Note: the function definitions for Questions 1-5 will go in this file as well. This is the file (the entire program) you need to submit to the CodeRunner auto-grader on Moodle.**

The menu will run on a loop, continually offering the user six options until they opt to quit.

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 no books are saved to the database due to wrong file name, then print the following message:

  ```
  No books saved to the database
  ```

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: <numberOfUsers>
     ```

   - If no books are saved to the database due to wrong file name, then print the following message:

     ```
     No users saved to the database
     ```

3. Display library
   - Call your `printAllBooks` function.

4. Get number of books reviewed by a user
   - Prompt the user for a username.
   - Pass the username to your `getUserReadCount` function
   - If the user exists in the system, print the result in the following format:

     ```
     <name> rated <numBookRead> books
     ```

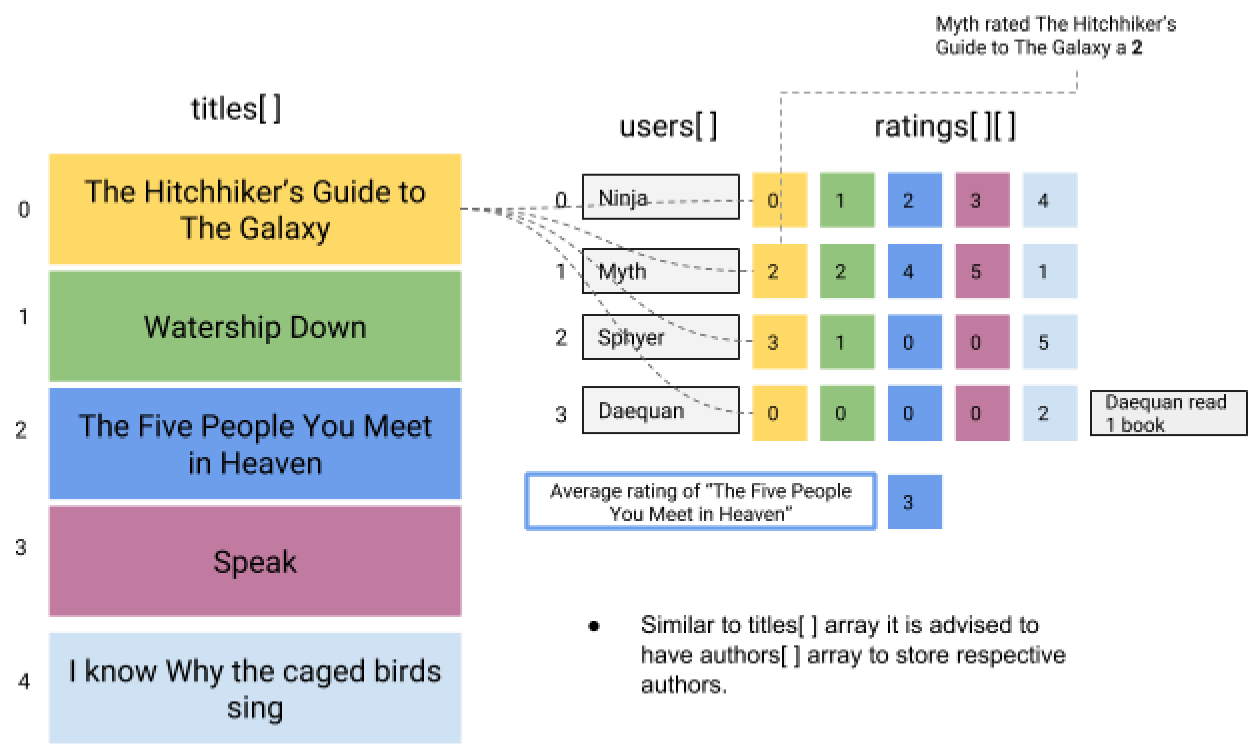
5. Get average rating for a title
   - Prompt the user for a title.
   - Pass the title to your `calcAvgRating` function
   - If the title exists in the database, print the result in the following format:

     ```
     The average rating for <bookTitle> is <value>
     ```

   `Note: <value>` is a double with 2 decimal points.

6. Quit
   - Print “`good bye!`” before exiting

Visualization of various elements in HW6



Myth rated The Hitchhiker's Guide to The Galaxy a **2**
titles[ ]
users[ ]
ratings[ ][ ]
0 The Hitchhiker's Guide to The Galaxy
1 Watership Down
2 The Five People You Meet in Heaven
3 Speak
4 I know Why the caged birds sing
0 Ninja 0 1 2 3 4
1 Myth 2 2 4 5 1
2 Sphyer 3 1 0 0 5
3 Daequan 0 0 0 0 2
Daequan read 1 book
Average rating of "The Five People You Meet in Heaven" 3

- Similar to titles[ ] array it is advised to have authors[ ] array to store respective authors.

**Sample output:** (user can select 1, 2, 3, 4, 5 or 6. The user input in the **bold**)

```
Select a numerical option:
======Main Menu=====
1. Read book file
2. Read user file
3. Print book list
4. Find number of books user rated
5. Get average rating
6. Quit
1
Enter a book file name:
wrongfile.txt
No books saved to the database

Select a numerical option:
======Main Menu=====
1. Read book file
2. Read user file
3. Print book list
```

```
4. Find number of books user rated
5. Get average rating
6. Quit
3
No books are stored

Select a numerical option:
======Main Menu=====
1. Read book file
2. Read user file
3. Print book list
4. Find number of books user rated
5. Get average rating
6. Quit
4
Enter username:
ioana
ioana does not exist in the database

Select a numerical option:
======Main Menu=====
1. Read book file
2. Read user file
3. Print book list
4. Find number of books user rated
5. Get average rating
6. Quit
1
Enter a book file name:
books.txt
Total books in the database: 50

Select a numerical option:
======Main Menu=====
1. Read book file
2. Read user file
3. Print book list
4. Find number of books user rated
5. Get average rating
6. Quit
2
Enter a rating file name:
ratings.txt
cynthia...
```

```
diane...
joan...
( . . . truncated . . . )
raymond...
adam...
johnny...
Total users in the database: 86

Select a numerical option:
======Main Menu=====
1. Read book file
2. Read user file
3. Print book list
4. Find number of books user rated
5. Get average rating
6. Quit
3
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
I Know Why the Caged Bird Sings by Maya Angelou
Thirteen Reasons Why by Jay Asher
Foundation Series by Isaac Asimov
( . . . truncated . . . )
Maus: A Survivor's Tale by Art Spiegelman
The Joy Luck Club by Amy Tan
The Lord of the Rings by J R R Tolkien

Select a numerical option:
======Main Menu=====
1. Read book file
2. Read user file
3. Print book list
4. Find number of books user rated
5. Get average rating
6. Quit
4
Enter username:
dorothy
dorothy rated 41 books

Select a numerical option:
```

```
======Main Menu=====
1. Read book file
2. Read user file
3. Print book list
4. Find number of books user rated
5. Get average rating
6. Quit
5
Enter book title:
The Princess Diaries
The average rating for The Princess Diaries is 2.10

Select a numerical option:
======Main Menu=====
1. Read book file
2. Read user file
3. Print book list
4. Find number of books user rated
5. Get average rating
6. Quit
5
Enter book title:
The Bourne Series
The average rating for The Bourne Series is 3.64

Select a numerical option:
======Main Menu=====
1. Read book file
2. Read user file
3. Print book list
4. Find number of books user rated
5. Get average rating
6. Quit
6
good bye!
```

**Grading Rubric**:

| Criteria | Points |
|---|---|
| Coderunner | 90 |
| Comments/Style | 10 |
| Total | 100 |
| Early submission bonus | + 5% or 2% |