CSCI 1300 Introduction to Programming
Instructor: Boese
Project #2
Due: **Tuesday**, December 9, by 10am.

Objectives

- Create class header files
- Create and use vector objects
- Read code written by someone else and understand it

Recommendation System

This project builds on your Homework #10. If you did not get it to work, you will need to use the example answer key for Homework 10. (see Moodle, available Dec 1 by 9pm). Please refer back to Assignment #10 for details that apply to this project.

Rating System

Rating	Meaning
0	Haven't read it
-5	Hated it!
-3	Didn't like it
1	ok - neither hot nor cold about it
3	Liked it!
5	Really liked it!

Your goal for Project #2 is to get to the following output:

```
Reading in book list from file: books.txt
55 books read in. Closing book list file.
Reading in user list from file: ratings.txt
32 users read in. Closing user file.
Finding recommendations for user: Liz
Most similar user: moose

We recommend for you the following books:
The Hitchhiker's Guide To The Galaxy by Douglas Adams
Watership Down by Richard Adams
Foundation Series by Isaac Asimov
Ender's Game by Orson Scott Card
The Hunt for Red October by Tom Clancy
Neuromancer by William Gibson
Lord of the Flies by William Golding
To Kill a Mockingbird by Harper Lee
The Lion the Witch and the Wardrobe by C S Lewis
Nineteen Eighty-Four (1984) by George Orwell
Harry Potter Series by J.K. Rowling
The Lord of the Rings by J R R Tolkien
The Hobbit by J R R Tolkien
The War Of The Worlds by H G Wells
```

Assignment Details:

Part 1 – Create Header Files

Modify your `Book.cpp` and `User.cpp` files to make use of header files. Make sure you include the `#ifndef` and `#define` as well and do not use "`using namespace std`" inside the header files.

You can compile-only by following the slides "Compiling" added to the Classes ppt slides.

- Convert `Book.cpp` to have `Book.h` and `Book.cpp`, and `User.cpp` to have `User.h` and `User.cpp`
    - Define the class in the respective header files, and put the function definitions in the .cpp files.
    - Use `#ifndef` and `#define` in both `.h` files
    - Do not use `using namespace std` in the `.h` files, associate using `std::`
- Compile the Book files:
  ```
  g++ -c Book.cpp
  ```
  You should now have an additional file in your directory: Book.o
- Compile the User files:
  ```
  g++ -c User.cpp
  ```
  You should now have an additional file in your directory: User.o
- Download the tester to help see if your code may be working
  (Note: just because the test driver program works, does not mean you did Part 1 correctly. This is just to help you start testing).
  ```
  g++ Book.o User.o TestPart1Driver.cpp
  ```

Part 2

Download the Library files. Download the new `Library.cpp and Library.h` files and ensure it compiles.

- Download `Library.h` and `Library.cpp`
- Compile just the Library files
  ```
  g++ -c Library.cpp
  ```
- Download the driver `DriverPart2.cpp`
- Download the books and users files: `books.txt`, `ratings.txt`
- Compile and run the driver program:
  ```
  g++ *cpp
  ./a.out
  ```
  You should see the output of all the users and all the books.

Part 3 – Find most similar user

The purpose for part 3 is to determine which of the users in the list is most similar to the current user. We want this information for Part 4 where we print out recommendations based on the most similar user.

You can calculate how similar two users are by treating each of their ratings as a mathematical vector and calculating the dot product of these two vectors. (Remember that the dot product is just the sum of the products of each of the corresponding elements.)

For example, suppose we had 3 books and

Terry rated them	[5, 3, 1]
Bob rated them	[5, 1, 5]
Tracey rated them	[1, 5, 3]
Kalid rated them	[1, 3, 0]

The similarity between

- Terry and Bob is calculated as: (5 x 5) + (3 x 1) + (1 x 5) = 25 + 3 + 5 = 33
- Terry and Tracey is: (5 x 1) + (3 x 5) + (1 x 3) = 5 + 15 + 3 = 23
- Terry and Kalid is (5 x 1) + (3 x 3) + (1 x 0) = 5 + 9 + 0 = 14

Once you have calculated the pair-wise similarity between Terry and every other customer, you can then identify whose ratings are most similar to Terry's. In this case Bob is most similar to Terry.

Algorithm to get the most similar user:

```
SET maxSimilarity to 0
SET mostSimilarUser to unknown
LOOP through each user in the library's list
    IF that user is not the user we are comparing to
        IF that user has a higher similarity score than the maxSimilarity value
            Save the score and the user
```

To determine similarity between two users, create a method inside the `User` class:

```
/**
 * Function to determine how similar another user is to this one
 * @param otherUser the other user to check similarity with
 * @return similarity value based on the dot product calculation
 */
int User::getSimilarity(User otherUser)
{
    // Part 3
}
```

The Library class calls this function. You will need to add the following code inside the Library file and fill in the function below that makes use of your getSimilarity function in the User class:

```
/**
 * Return the most similar user
 * @param currentUser the user we want to find a similar user for
 * @return
 */
User Library::getMostSimilarUser(User currentUser)
{
    double bestSimilarity = 0;
    User bestUser; // keep track of the user with the highest score

    // ---------------------------------------------------
    // PART 3
    // ---------------------------------------------------
    // loop through each user and calculate the similarity score
    //   by calling the getSimilarity method from the User class


    // ---------------------------------------------------
    return bestUser;
}
```

For this part, use the ProjectDriver.cpp file. You will need 3 command-line arguments such that your command-line arguments are as follows:

```
books.txt  ratings.txt  currentUserRatings.txt
```

The third command-line argument contains our current user's id and ratings. This is the user will be comparing to find the most similar user.

You should now be able to output the most similar user in the Library class:

```
Most similar user: andrew
```

- Download ProjectDriver.cpp for use with Parts 3 and 4.
  You may not change it in any way, except for your own testing!
- Download currentUserRatings.txt
- Make the modifications as described above
- Compile and run the ProjectDriver program.

Part 4
Once we have a user who is most similar to our current user, we can recommend books to our current user based on the similar user. Print out all books that the similar user rated as a 4 or 5 *that our current user has not read yet*.

For example, once you have calculated the pair-wise similarity between Terry and every other customer, you can then identify whose ratings are most similar to Terry's. In this case Bob is most similar to Terry, so we would recommend to Terry the top books from Bob's list *that Terry hasn't read yet*.

For example, from our example files if our current user is "Moose":

```
We recommend for you the following books:
The Hitchhiker's Guide To The Galaxy by Douglas Adams
Watership Down by Richard Adams
Foundation Series by Isaac Asimov
Ender's Game by Orson Scott Card
The Hunt for Red October by Tom Clancy
Neuromancer by William Gibson
Lord of the Flies by William Golding
To Kill a Mockingbird by Harper Lee
The Lion the Witch and the Wardrobe by C S Lewis
Nineteen Eighty-Four (1984) by George Orwell
Harry Potter Series by J.K. Rowling
The Lord of the Rings by J R R Tolkien
The Hobbit by J R R Tolkien
The War Of The Worlds by H G Wells
```

Inside the Library files, add a function called `printRecommendations`.

```
/**
 * Print out the books rated by the user with a 4 or 5 rating that the
 *   current user has not read yet
 * @param mostSimilarUser   user with the most similarly rated books to current user
 * @param currentUser the current user we are making recommendations for
 */
void Library::printRecommendations(User mostSimilarUser, User currentUser)
{
   cout << "We recommend for you the following books:" << endl;
      // your code here
}
```

EXTRA CREDIT CHALLENGES
Some extra credit is available, but you will not get any extra credit unless you have 100% of the actual assignment working correctly.

[3 pt] **XC-1**: Copy the ProjectDriver.cpp file to a new file named **ProjectDriverXC1.cpp** so that it is interactive. A completed challenge will do all of the following:

- Ask the user for their ID, and read that in
- Find the user in the library that corresponds to their ID
- Provides a list of recommended books for that user based on that user's ratings

The interaction should begin after all of the regular project assignment output.

[3 pt] **XC-2**: Copy your **ProjectDriverXC1.cpp** file to a new a new file named **ProjectDriverXC2.cpp**
Give the user the option to enter ratings for books they haven't rated yet (books that have a zero rating). This requires you to have done XC-1 so that you know which userid is theirs. You will need to then modify the file (re-write the ratings back to the ratings file, e.g., ratings.txt) with all the ratings to include their newly rated books as well.

Additional Requirements

- The name of the files must be called **Book.cpp**, **Book.h** and **User.cpp, User.h** and **Library.h**, **Library.cpp, ProjectDriver.cpp** (and **ProjectDriverXC1.cpp, ProjectDriverXC2.cpp** if you did the extra credit)
  Do not submit TestPart1Driver.cpp, DriverPart2.cpp – these are just for you to play with.
- You must zip these file up to submit it in Moodle as **Firstname_Lastname_Project2.zip**!
- Comments at the top of EACH of your class files
  - Your name
  - Date
  - Project #2
  - Brief description of the file (one or two lines max)
- Example files are posted on Moodle for you to download. TEST WITH ADDITIONAL DIFFERENT FILES!
- The output must match exactly to the examples provided (given appropriate inputs).
- Program must be written in C++ and submitted in Moodle.