CS216: Introduction to Software Engineering Techniques (Spring, 2022)
Project Assignment 1

(100 points)

Today's Date: Friday, February 11

Due Date: Friday, February 25

(You should include your name and section number as part of the comments for your project assignment!!!)

Problem Statement

The C++ program that you will write is to solve the similar problem as in Lab4 assignment (with some extra part), but in C++ programming language instead of Shell Script.

Websites like IMDB (which stands for **Internet Movie Data Base**) maintain all sorts of info about movies, the actors etc. If you search for a movie on the website, a web page showing information about the movie is displayed. It also shows all the actors in it. If you click on the hypertext link for an actor, you are taken to the actor's web page and can find all the information about him/her. i.e., names of movies in which the actor has acted, some other information. This assignment should give you some insight into the working of such websites. In order to complete this project you need some basic movie/actor data. Rather than giving you a huge database like IMDB uses, you are provided with a relatively small data file named actor_movies_long.txt, which is exactly the same as the text file you use in Lab5.

The input file you need to use to test your Project 1 is named actor_movies_long.txt, you can either copy it from Lab6 directory, or download it from the following link:

```
$ curl -O https://www.cs.uky.edu/~yipike/CS216/actor_movies_long.txt
```

The format of this file is very simple. Each line represents one actor/actress name and a partial listing of the movies, which this actor/actress was in. The format of each line of this file is:

`Actor/actress Name; movie1;  movie2; movie3; …; movieN;`

(Note that an actor/actress name may contain blank space, however each full name ends with "**;**", then followed by a list of movie titles separated by "**;**", and a movie title may contain "**,**" or "**:**" as part of the title, however, each complete movie title ends with "**;**").

After reading the data from the above input file, and storing the data to an object of IMDB class (which you are going to define for this project), your program should allow the user to repeatedly choose from the following main menu until the user enters "Q" or "q" to quit the program:

```
This application stores information about Actors and their
Movies, please choose your option (Enter Q or q to quit):
1. Actors in Movies
2. Actors and co-actors
```

If the user chooses option 1:

Ask the user to type the titles of two movies;

Are both of the titles valid (it is valid if the movie title that user inputs, matches a movie title in the file named actor_movies_long.txt; otherwise it is invalid)?

a) If yes, display two matched movie titles, then continue performing one of three "searches" and print the result.
b) If no, print an appropriate error message and quit.

Continue performing one of three "searches" in movies:

Repeatedly display a sub-menu to let the user choose an option (enter Q or q to quit the sub-menu, back to the main menu):

1.if option is A (or a), print all the actors in either of the two movies;
2.if option is C (or c), print all the common actors in both of the movies;
3.if option is O (or o), print all the actors who are in only one movie, but not in both.
4.if option is Q (or q), back to the main menu
5.if other option, display the message of invalid choice.

If the user chooses option 2:
Ask the user to type an actor's name;
Is the actor's name valid (it is valid if the name appears in the file named actor_movies_long.txt; otherwise it is invalid)?

a) If yes, print a well-formatted message about all of that actor's co-actors. That is, for each movie, which the user-input actor was in, print a message containing all of the co-actors who appeared in the same movie, then back to the main menu.
b) If no, print an appropriate error message, then back to the main menu.

As we learned from Math class, the three "searching" operations above can be described as the following mathematic operations (note that we use mathematic operations here for understanding purpose):
Searching for option A can be represented as **`R union S: (R⋁S)`**;
Searching for option C can be represented as **`R intersection S: (R⋀S)`**;
Searching for option O can be represented as **`R symmetric_difference S: (R⋁S)-(R⋀S)`**.
Where **`R`** is the set of actors in the first movie, and **`S`** is the set of actors in the second movie.
And the co-actors of the actor, say **`K`**, is the `SET` of actors who are in at least one movie with **`K`**.
More detailed requirement for validation of movie titles and actor names:

- Since it is not flexible to ask the user to exactly match the complete movie title, your program should take the user-input movie title, say movieName, which is a string object, to match a movie in the input file: it is defined as "matched" if a movie title in the input file contains movieName as a substring and it performs case insensitive matching. For example, if user types "got mail", your program should find the matched movie title of "You've Got Mail (1998)". If there are more than one matched movie titles from the input file, your program only need to pick the first matched one.

- Since it is common to have same first name, to validate an actor name, your program can simply apply exactly match, which means the user needs to type the actor name exactly as that of the input file. To make it simple, this project considers case sensitive for the actor names, for example, if the user types "tom hanks", your program can simply report this name is not valid.

How do I do that??

What is an appropriate data structure for this project? There are more than one options, however, for this project, **you are required to use the map class and the set class from C++ standard library to store the information about movies and actors from the file named actor_movies_long.txt, into the private data members of IMDB class.**

The following shows the declaration of the class named IMDB, and you will download the header file and provide the complete definition of this class in Lab5.

```
class IMDB
{
    public:
        IMDB();         // default constructor

        // insert a pair<actorName, the set of movieTitle of this actor>
        void insert_an_actor(string actorName, set<string>
movieTitles);

        // insert a pair <movieTitle, the set of actor names in this movie>
        void insert_a_movie(string movieTitle, set<string>
actorNames);

        // use passing-in parameter, movieTitle, as the pattern to match
        // one existing movie in the map
        // return the matched movie title from the map
        // otherwise return empty string
        string matchExistingMovie(string movieTitle) const;

        // check if a movieTitle does exist in the map
        // return true if it does; otherwise return false
        bool isExistingMovie(string movieTitle) const;

        // check if an actorName does exist in the map
        // return true if it does; otherwise return false
```

```
            bool isExistingActor(string actorName) const;

            // return a set of movie titles which actorName is in
            // if the passing-in parameter: actorName, is not in the map,
            //             display message and return an empty set
            set<string> find_movies_for_an_actor(string actorName)
const;

            // return a set of actor names which are all in the movieTitle
            // if the passing in-parameter: movieTitle is not in the map,
            //             display message and return an empty set
            set<string> find_actors_in_a_movie(string movieTitle)
const;

            // you are allowed to add other member functions if you need

      private:
            // map of <movie title, set of actors in this movie>
            map<string, set<string> > movies_db;

            // map of <actor name, set of movies this actor is in>
            map<string, set<string> > actors_db;

};
```

Provide your complete definition of this class named IMDB, in imdb.cpp file. Then compile the source code:

```
$ g++ PA1.cpp imdb.cpp -o Project1
```

After passing the compilation, please download the following sample output file to test running your program, and also think about how to design your own testing cases:

http://www.cs.uky.edu/~yipike/CS216/PA1Sample_imdb.pdf

After testing your program with sample testing cases and testing cases you design by yourself, please follow the following instruction to prepare your submission:

Electronic program submission:

Zip the following files:

1. *.cpp and *.h – the C++ source files and header file(s), zip all of them.
2. actor_movies_long.txt – the input text file you download

Names to use:
Name the zip file Project1.zip

Use the `zip` program on your Virtual Machine, not `tar` or `gzip`. Use the zip program directly in the directory where the source files are. For example, if the source **`.cpp`** files and **`.h`** file(s) are in directory `PA1` under your `CS216` directory:

```
$ cd CS216
$ cd PA1
$ zip Project1.zip *.cpp *.h *.txt
```

DO not zip either from your home directory or from CS216 directory.
This creates a subdirectory when your zip file is unzipped, and will cause the grader extra work. When you cause the grader extra work, you lose points.

Submission:
Open the link to Canvas page (https://www.uky.edu/canvas), and log in to your account using your linkblue ID and password. Please submit your file (`Project1.zip`) through the submission link for "**`Project 1`**".

(Late assignment will be reduced 10% for each day that is late. The assignment will not be graded (you will receive zero) if it is more than 5 days late. Note that a weekend counts just as regular days. For example, if an assignment is due Friday and is turned in Monday, it is 3 days late.)

Always read the grading sheet for each project assignment. It lists typical errors. Check for these errors before submitting your source code. **Please note that your C++ program must compile in order to be graded.** If your program cannot pass the compilation, you will get 0 point.
(The grading sheet is on the next page.)

Academic Honesty:
All assignments in class are individual work. All work submitted as part of the class must be your own. You may not share work (whether from quizzes, lab assignments, or project assignments), nor may you use code provided to you by others, except for your instructor. You are allowed to use the source code provided by the instructor of this course only.

Grading Sheet for Programming Assignment 1

Total: 100 points.

C++ program must compile in order to be graded	Points	Deducted Points
C++ Program	75	
Check the command line argument	2	
File open errors detected correctly	2	
Read data from the input file, and store the information to the `IMDB` object correctly	5	
Provide the correct definition of IMDB class	6	
Repeatedly allow the user to choose an option from the main menu, until the user enters Q or q to quit	3	
Option 1, check if the titles for two movies which the user enters, are both valid and handle accordingly	3	
Repeatedly allow the user to choose an option from the sub-menu of three “searching” operations, until the user enters Q or q to quit the sub-menu and back to the main menu	3	
Option A (or a) displays the correct result	10	
Option C (or c) displays the correct result	10	
Option O (or o) displays the correct result	10	
Handle the invalid option from the sub-menu and main menu	2*2 = 4	
Option 2, check if the name of the actor which the user enters, is valid or not	5	
Option 2, generate the correct well-formatted message about co-actors with the actor the user input: in which movie, all the co-actors, the user input name should not be in the co-actors list	10	
Quit the program correctly	2	
Miscellaneous errors, or did not follow the directions in the program assignment, examples:	10	
-3 did not provide movie_actors_long.txt	3	
-3 did not zip file or used tar or gzip instead	3	
-2 created subdirectory when unzipped	2	
-2 wrong names	2	
There may be other errors in this category		
Coding Style	15	
Provide user-friendly interface	3	
Lay out your program in a readable style	3	
C++ program comments	9	
-9 non		
-7 contains only the purpose of the program		
-5 only a few, no comments for functions		
Your Score		