CS 201, Fall 2015

Homework Assignment 3

Due: 23:59, December 30 (Wednesday), 2015

In this homework assignment, you are supposed to implement a movie database by using linked lists. In this movie database, for each movie, you will have an entry in which you keep:

1. The movie title,
2. The day, month, and year the movie is/was released to theaters,
3. The first name and last name of the director,
4. A list of its cast. For each actor/actress in the cast, you should keep:
    - The first and last name of an actor/actress, and
    - The title of the role s/he played in the movie.

In your implementation, you **MUST** keep the movie entries in a doubly linked list. For each of these movie entries, you **MUST** use another linked list to keep its cast; this **MUST** be a sorted singly linked list where the actors/actresses **MUST** be ordered alphabetically (first according to the last name, then according to the first name).

Your system will have the following functionalities; the details of these functionalities are given below:

1. Add a movie
2. Remove a movie
3. Add an actor/actress to the cast of a movie
4. Remove an actor/actress from the cast of a movie
5. Show the list of movies
6. Show detailed information about a particular movie
7. Query the roles that were played by a particular actor/actress
8. Query the movies which are/were released in a particular year

Add a movie: This function adds an entry to the movie database for a given movie whose title, director's name (first name + last name), and release date (day + month + year) are specified as parameters. In this function, the cast is not specified; the actors/actresses that play in the movie will be added later. In this system, the titles of the movies are unique (i.e., no remakes of old movies are considered under the same title). Thus, if the user attempts to add a movie with an existing title, you should not allow this operation and give a warning message.

Remove a movie: This function removes a movie entry, whose title is specified as a parameter, from the movie database. If this movie does not exist in the database (i.e., if there is no movie with the specified title), you should not allow this operation and give a warning message.

Add an actor/actress: This function adds an actor/actress to the cast of a movie. For that, the title of the movie to which the actor/actress is added, the actor/actress' name (first name + last name), and the title of the role played by the actor/actress are specified as parameters. In this function, you should consider the following issues:

- If the movie with a specified title does not exist in the database, you should not allow the operation and give a warning message.
- In this system, all actor/actress' names (the first name together with the last name) are unique within the same cast. Thus, if the user attempts to add an actor/actress with an existing name, you

should not perform the operation and give a warning message. Here note that an actor/actress can play in different movies.

- The cast should be in alphabetically ascending order according to the last names of actors/actresses; if you have more than one entry with the same last name, you should also consider the first names.

Remove an actor/actress: This function removes an actor/actress from the cast of a movie. For that, the title of the movie from which the actor/actress is removed and an actor/actress' name (first name + last name) are specified as parameters. If the movie with a specified title does not exist in the database, you should not allow the operation and give a warning message. Similarly, if the actor/actress is not in the cast of the specified movie, you should not allow the operation and give a warning message. Note that after calling this function, the cast should remain sorted in ascending order.

Show the list of movies: You should list all movie entries in the movie database on the screen in the following format. If the database does not contain any movie, you should display `---none---`.

```
Title, release year, director name    (for the 1st movie)
Title, release year, director name    (for the 2nd movie)
Title, release year, director name    (for the 3rd movie)
...
```

Show detailed information about a particular movie: You should display all of the information about a movie whose title is specified as a parameter. The output should be in the following format where the actor/actress' names should be sorted in alphabetically ascending order. If the movie with a specified title does not exist in the database, you should display `---none---` after the title.

```
Title
Release day / release month / release year
Director name
Actor/actress name, title of his/her role  (for the 1st actor/actress)
Actor/actress name, title of his/her role  (for the 2nd actor/actress)
Actor/actress name, title of his/her role  (for the 3rd actor/actress)
...
```

Query the roles that were played by a particular actor/actress: You should list all roles played by the actor/actress whose name (first name + last name) is specified as a parameter. The output should include the role, movie title, and the year, and should be in the following format. Note that if the actor/actress did not play a role in any movie, you should display `---none---` after the name of the actor/actress.

```
Actor/actress name
Role, movie title, release year       (for the 1st role)
Role, movie title, release year       (for the 2nd role)
Role, movie title, release year       (for the 3rd role)
...
```

Query the movies which are/were released in a particular year: You should list all movies released in the year which is specified as a parameter. The output should include the title of the movies, the director names, and release dates, and should be in the following format. Note that if there is no movie in the specified year, you should display `---none---` after the release year.

```
Release year
Movie title, director name, release day/release month (for the 1st role)
Movie title, director name, release day/release month (for the 2nd role)
Movie title, director name, release day/release month (for the 3rd role)
...
```

Below is the required public part of the `MovieDatabase` class that you must write in this assignment. The name of the class **MUST** be `MovieDatabase`, and **MUST** include these public member functions. We will use these functions to test your code. The interface for the class must be written in a file called `MovieDatabase.h` and its implementation must be written in a file called `MovieDatabase.cpp`. You can define additional public and private member functions and data members in this class. You can also define additional classes in your solution.

```
class MovieDatabase {
public:
     MovieDatabase();
     ~ MovieDatabase();


     void addMovie(const string movieTitle, const string directorFirstName,
                   const string directorLastName, const int releaseDay,
                   const int releaseMonth, const int releaseYear);

     void removeMovie(const string movieTitle);

     void addActor(const string movieTitle, const string actorFirstName,
                   const string actorLastName, const string actorRole);

     void removeActor(const string movieTitle, const string actorFirstName,
                      const string actorLastName);

     void showAllMovies();

     void showMovie(const string movieTitle);

     void showActorRoles(const string actorFirstName,
                         const string actorLastName);

     void showMovies(const int releaseYear);

     // ...
     // you may define additional member functions and data members,
     // if necessary.

};
```

IMPORTANT NOTES ABOUT IMPLEMENTATION:

Do not start your homework before reading these notes!!!

1. For this assignment, you must use your own implementation of linked lists. In other words, you cannot use any existing linked list code from other sources such as the list class in the C++ standard template library (STL). Moreover, you are not allowed using array-based implementation or data structures such as `vector` from the standard library. However, you can adapt the linked list codes in the Carrano book. You will get no points if you do not use linked lists as indicated.

2. You ARE NOT ALLOWED modifying the given parts of the MovieDatabase class. However, if necessary, you may define additional data members and member functions in this class, or use additional classes. However, you ARE NOT ALLOWED using any global variables or any static local variables.

3. Your code must not have any memory leaks. You will lose points if you have memory leaks in your program even though the outputs of the operations are correct.

4. Your implementation should consider all names (movie titles, first names, last names, and actor/actress' roles) case insensitive. For example, the movie titles "Take It All" and "take IT all" should be considered as the same.

NOTES ABOUT SUBMISSION:

1. In this assignment, you must have separate interface and implementation files (i.e., separate .h and .cpp files) for your class. The file names should be "`MovieDatabase.h`" and "`MovieDatabase.cpp`". You should also submit other .h and .cpp files if you implement additional classes. We will test your implementation by writing our own main function. Thus, you should not submit any function that contains the main function.

   Although you are not going to submit it, we recommend you to write your own driver file to test each of your functions. However, you SHOULD NOT submit this test code (we will use our own test code).

2. You should put your "`MovieDatabase.h`" and "`MovieDatabase.cpp`" (and additional .h and .cpp files if you implement additional classes) into a folder and zip the folder (in this zip file, there should not be any file containing the main function). The name of this zip file should conform the following name convention: secX-Firstname-Lastname-StudentID.zip where X is your section number.

   The submissions that do not obey these rules will not be graded.

3. **Then, before 23:59 on December 30, you need to send an email with a subject line CS 201 HW3 to either Troya Cagil Koylu or Can Fahrettin Koyuncu, by attaching this zipped file containing only your header and source codes (but not any file containing the main function).**

   No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You are free to write your programs in any environment (you may use either Linux or Windows). On the other hand, we will test your programs on "`dijkstra.ug.bcc.bilkent.edu.tr`" and we will expect your programs to compile and run on the `dijkstra` machine. If we could not get your program properly work on the `dijkstra` machine, you would lose a considerable amount of points. Therefore, we recommend you to make sure that your program compiles and properly works on "`dijkstra.ug.bcc.bilkent.edu.tr`" before submitting your assignment.

This homework will be graded by your TAs Troya Cagil Koylu (troya at cs bilkent edu tr) and Can Fahrettin Koyuncu (koyuncu at cs bilkent edu tr). Thus, you may ask your homework related questions directly to them.