CS 201, Fall 2016

Homework Assignment 3

Due: 23:59, Jan 9 (Monday), 2017

In this homework, you will implement a jukebox that contains multiple tracks using linked lists. In this jukebox, for each track, you will have an entry, in which you keep:

1. The track title,
2. The year that the track is/was released, and
3. A list of its musicians. For each musician, you should keep:
    - The first and last name of the musician and
    - The role of the musician; a musician could be singer, composer, or lyricist.

In your implementation, for the jukebox, you **MUST** keep its track entries in a linked list. For each of these track entries, you **MUST** use another linked list to keep its musicians.

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

1. Add a track
2. Remove a track
3. Add a musician to a track
4. Remove a musician from a track
5. Show the list of tracks
6. Show detailed information about a particular track
7. Query a musician

Add a track: This function adds an entry to the jukebox for a given track whose title and release year are specified as parameters. In this function, the musician list is not specified; the musicians of the track will be added later. In this system, the titles of the tracks are unique (i.e., no remakes of old tracks are considered under the same title). Thus, if the user attempts to add a track with an existing title, you should not allow this operation and give a warning message.

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

Add a musician to a track: This function adds a musician to the musician list of a track. For that, the title of the track to which the musician is added, the musician name (first name + last name), and the role taken by the musician are specified as parameters. The role of a musician could be singer, composer or lyricist. In this function, you should consider the following issues:

- If the track with a specified title does not exist in the jukebox, you should not allow the operation and give a warning message.
- In this system, all musician names are unique within the same track. Thus, if the user attempts to add a musician with an existing name (to the same track), you should not perform the operation and give a warning message. Here, note that a musician can take roles in different tracks (e.g., a musician could be a singer in one track and a composer in two different tracks; but s/he cannot be both a singer and a composer in the same track). You may assume that the names are case insensitive.

Remove a musician from a track: This function removes a musician from a track. For that, the title of the track from which the musician is removed and the musician name (first name + last name) are specified as parameters. If the track with a specified title does not exist in the jukebox, you should not allow the operation and give a warning message. Similarly, if the musician is not in the musician list of the specified track, you should not allow the operation and give a warning message.

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

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

Show detailed information about a particular track: You should display all of the information about a track whose title is specified as a parameter. The output should be in the following format. If the track with a specified title does not exist in the jukebox, you should display `---none---` after the first line.

```
Title, release year
Musician name, role (for the 1st musician)
Musician name, role (for the 2nd musician)
Musician name, role (for the 3rd musician)
. . .
```

Query a musician: You should list all tracks in which a musician takes a part. In this function, the name (first name + last name) of a musician is specified as a parameter. The output should include the role, the track title, and the release year, and should be in the following format. Note that if the musician did not take a part in any track, you should display `---none---` after the first line.

```
Musician name
Role, track title, release year (for the 1st track)
Role, track title, release year (for the 2nd track)
Role, track title, release year (for the 3rd track)
. . .
```

Below is the required public part of the `JukeBox` class that you must write in this assignment. The name of the class **must** be **`JukeBox`**. The interface for the class must be written in a file called `JukeBox.h` and its implementation must be written in a file called `JukeBox.cpp`. Your class interface file ("`JukeBox.h`") should contain the following member functions. If necessary, you may also define additional public and private member functions and data members in this class. You can also define additional classes in your solution. On the other hand, you are not allowed to delete any of the given functions.

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

    void addTrack( string trackTitle, int releaseYear );
    void removeTrack( string trackTitle );

    void addMusician( string trackTitle, string musicianFirstName,
                      string musicianLastName, string musicianRole );
    void removeMusician( string trackTitle, string musicianFirstName,
                         string musicianLastName );

    void showAllTracks();
    void showTrack( string trackTitle );
    void showMusicianRoles( string musicianFirstName,
                            string musicianLastName );

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

NOTES ABOUT IMPLEMENTATION:

1. You ARE NOT ALLOWED to modify the given parts of the header file. You MUST use linked lists in your implementation. You will get no points if you use array-based implementation or data structures such as `vector` from the standard library. However, if necessary, you may define additional data members and member functions. Moreover, you ARE NOT ALLOWED to use any global variables or any static local variables.
2. You may use the C++ codes that are given in your textbook or the class slides. But you ARE NOT ALLOWED to use any other list implementations; for example, you cannot use the list class defined in another book or you cannot use any functions in the C++ standard template library (STL).
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 (track titles, musician first names, musician last names, and musician roles) case insensitive. For example, the track titles "Take It All" and "take IT all" should be considered as the same track.

NOTES ABOUT SUBMISSION:

1. This assignment is due 23:59 on Monday, Jan 9th, 2017. You should e-mail your homework to **ALI YESILYAPRAK (ali.yesilyaprak at bilkent edu tr)** before the deadline. 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.
2. 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 "`JukeBox.h`" and "`JukeBox.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).
3. You should put your "`JukeBox.h`" and "`JukeBox.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.
4. **Then, before 23:59 on January 9, you need to send an email with a subject line CS 201 HW3 to Ali Yesilyaprak, 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.
5. 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 "`dijkstra2.ug.bcc.bilkent.edu.tr`" and we will expect your programs to compile and run on the `dijkstra2` machine. If we could not get your program properly work on the `dijkstra2` machine, you would lose a considerable amount of points. Therefore, we recommend you to make sure that your program compiles and properly works on "`dijkstra2.ug.bcc.bilkent.edu.tr`" before submitting your assignment.
6. This homework will be graded by your TA **ALI YESILYAPRAK (ali.yesilyaprak at bilkent edu tr).** Thus, you may ask your homework related questions directly to him.