ASSIGNMENT2

Subject : Linked List
TAs: Merve Ozdes
Programming Lanuage: C++
Due Date: 03.12.2020 23:59

1 Introduction

In this homework, you will implement a library MovieShare system by using linked lists. Information about users and movies will be stored in this library system. The users will be able to check out movies from the library and will be able to return them to the library. In this system,there MUST be three linked lists for users. These are a linked list to store users and a linked list for each user to store movies checked out by that user, and a linked list that stores movies not checked out by any user. The features of these lists are given followings. For each users, the system stores an id, title, and a list of movies that she/he checked out. Each movie is represented by its id, title, and genre. The system MUST use a circular doubly linked list with no dummy head node to store the users, and for each user, a circular singly linked list with no dummy head node to store the movies checked out by this user. There MUST be an additional circular singly linked list with no dummy head node that stores the movies that are not checked out by any user. All lists can be unsorted. You can assume that there is a single copy of each movie. The Library system you will design will have the following functions; The details of these functions are given below:

- Add a movie
- Delete a movie
- Add a user
- Delete a user
- Check out a movie by a user
- Return a movie
- Show the list of all movies
- Show detailed information about a particular movie
- Show detailed information about a particular user

2 Background

A linked list is a sequence of data structures, which are connected together via links.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Linked list can be visualized as a chain of nodes, where every node points to the next node.

As shown in Figure 1, following are the important points to be considered.

- Linked List contains a link element called first.
- Each link carries a data field(s) and a link field called next.
- Each link is linked with its next link using its next link.
- Last link carries a link as null to mark the end of the list.

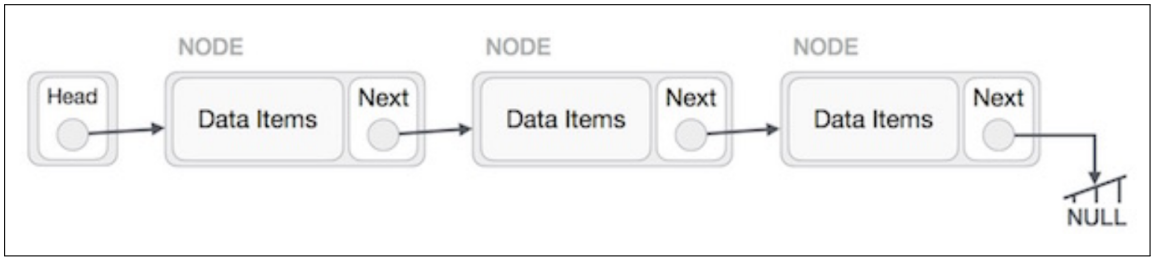

Figure 1: Representation of Linked list

Following are the various types of linked list.

Simple Linked List: Item navigation is forward only.

Doubly Linked List: Items can be navigated forward and backward.

Circular Linked List: Last item contains link of the first element as next and the first element has a link to the last element as previous.

3 Problem Definition

3.1 PART 1

This part includes the following three functionalities:

- Add a movie
- Delete a movie
- Show the list of all movies

  When this part is graded, you are expected to display all of the movies in the system as being not checked out. Therefore, for this part, you may assume that all of the movies in the system are not checked out and there is no user information in the system. Thus, in this part, there MUST be one circular singly linked list with no dummy head node that stores the movies that are not checked out by any user. This list can be unsorted. You can assume that there is a single copy of each movie.

3.2 PART 2

In this part, you will complete the implementation of the entire library system (all of the 9 functionalities listed above).
The details of all of the functionalities are given below:
Add a movie: In the library system, a new movie can be added to the collection with movie ID, movie title and movie genre information. The movie ids must be unique positive integers. Movie IDs must be unique positive integers. Therefore, the system should check if the specified movie ID already exists (ie, another movie ID), and disallow the action if there is a movie and display a warning message.

Delete a movie: In the library system an existing movie specified with its movie id can be deleted. If the movie does not exist (i.e., if there is no movie with the specified id), the system should display a warning message. Note that it should be possible to delete movies which are checked out. If the movie to be deleted is already checked out by a user, it will be deleted from

the checkout linked list of the user. Otherwise, it will be deleted from the linked list of movies that are not checked out by any user.

Add a user: In the library system a new user can be added with user id and name. The user ids must be unique positive integers. Therefore, the system should check if the specified user ID already exists (ie, another user ID), and disallow the action if there is a user and display a warning message.

Delete a user: In the library system an existing user with user id can be deleted . If the user does not exist (i.e., if there is no user with the specified id), the system should display a warning message. If the user has checked out movies, those movies must be returned to the system as part of the delete operation for the user.

Checkout a movie by a user: The library system will allow to check out a particular movie by a particular user. For that, the movie id and the user id have to be specified. The system should first check whether or not this movie exists; if it does not, it should not allow the operation and display a warning message. The system should also check whether or not this user exists; if he/she does not, it should not allow the operation and display a warning message. If both the movie and the user exist and the movie is not already checked out, the check out operation must be performed. Note that a movie can be checked out by only one user. However, a user can check out multiple movies.

Return a movie: The library system will allow a user to return a movie. For that, the movie id has to be specified. If the movie does not exist, the system will display a warning message. It will also check whether or not this movie is checked out; if not, a warning message will be displayed, otherwise, the operation will be carried out.

Show the list of all movies: The library system will allow to display the list of all the movies. This list includes, for each movie, the movie id, movie title, and genre. For each movie that is checked out, id and name of the user who checked out the movie should also be displayed.

Show detailed information about a particular movie: The library system will allow to specify a movie id and display detailed information about that particular movie. This information includes the movie id, movie title, genre and whether or not the movie was checked out. If the movie is checked out, id and name of the user who checked out the movie should be displayed. If the movie does not exist (i.e., if there is no movie with the specified movie id), the system should display a warning message.

Show detailed information about a particular user: The library system will allow to specify a user id and display detailed information about that particular user. This information includes the user id, user name, and the list of movies checked out by this user including the movie id, movie title, and genre. If the user does not exist (i.e., if there is no user with the specified user id), the system should display a warning message.

3.3 Experiment

The required general part of the LibrarySystem class that you need to write in this assignment is given below. The Class you will be designing in the assignment must be named **LibrarySystem** and must contain the public member functions shown in Figure 2. The interface of the class must be written to a file called **LibrarySystem.h** and its application must be written to a file named **LibrarySystem.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.

Your system will take the input file which is commands.txt as parameters. The information in these input file is in mixed order.

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


    void addMovie( const int movieId, const string movieTitle, const int year );
    void deleteMovie( const int movieId );

    void addUser( const int userId, const string userName );
    void deleteUser( const int userId );

    void checkoutMovie( const int movieId, const int userId );
    void returnMovie( const int movieId );

    void showAllMovies();
    void showMovie( const int movieId );
    void showUser( const int userId );
};
```

Figure 2: LibrarySytem and it's functions

3.4 Input

There will be one input file which called commands.txt to creating linked list. commands.txt has include some commands such as:
<addMovie>, <deleteMovie>, <addUser>, <deleteUser>, <checkoutMovie>, <showUser>, <showMovie>, <showAllMovies>, <returnMovie>.

commands.txt file format is as shown below:
<**addMovie**><tab><**movieId**><tab><**movieTitle**><tab><**movieYear**>
<**deleteMovie**><tab><**movieId**>
<**addUser**><tab><**userId**><tab><**userName**>
<**deleteUser**><tab><**userId**>
<**checkoutMovie**><tab><**movieId**><tab><**userId**>
<**showUser**><tab><**userId**>
<**showMovie**><tab><**movieId**>
<**showAllMovies**>
<**returnMovie**><tab><**movieId**>

3.5 Outputs

You are expected to produce output.txt file according to commands.txt file. The format of commands.txt file is shown above. In Figure 3, output and commands file are shown. The output of each command is shown in the output file. Create your outputs according to the format in the output file.

```
addMovie        12      Adult World     2013
addMovie        9       Bolt    2008
addMovie        9       Shrek   2001
deleteMovie     4
deleteMovie     12
deleteMovie     12
deleteMovie     9
deleteMovie     4
addUser 12      Jon Snow
addUser 12      Jon Snow
addUser 2       Arya Stark
addUser 33      Sansa Stark
addUser 2       Arya Stark
addUser 99      Theon Greyjoy
deleteUser      12
deleteUser      666
deleteUser      99
deleteUser      12
deleteUser      33
deleteUser      2
deleteUser      444
addMovie        12      Fear    1990
addMovie        9       Bird Box        2018
addMovie        667     Midway  2019
addMovie        647     Tom Sawyer      1930
addUser 233     Robb Stark
addUser 1       Rhegal Targaryen
checkoutMovie   9       233
checkoutMovie   1000    233
checkoutMovie   12      11
checkoutMovie   12      1
checkoutMovie   9       233
showAllMovie
```

(a) commands.txt

```
===Movie Library System===

===addMovie() method test===
Movie 12 has been added
Movie 9 has been added
Movie 9 already exists

===deleteMovie() method test===
Movie 12 has not been checked out
Movie 12 has been deleted
Movie 9 has not been checked out
Movie 9 has been deleted

===addUser() method test===
User 12 has been added
User 12 already exists
User 2 has been added
User 33 has been added
User 2 already exists
User 99 has been added

===deleteUser() method test===
User 12 has been deleted
User 666 does not exist
User 99 has been deleted
User 12 does not exist
User 33 has been deleted
User 2 has been deleted
User 444 does not exist

===addMovie() method test===
Movie 12 has been added
Movie 9 has been added
Movie 667 has been added
Movie 647 has been added

===addUser() method test===
User 233 has been added
User 1 has been added

===checkoutMovie() method test===
Movie 9 has been checked out by User 233
Movie 1000 does not exist for checkout
User 11 does not exist for checkout
Movie 12 has been checked out by User 1
Movie 9 does not exist for checkout

===showAllMovie() method test===
Movie id - Movie name - Year - Status
647 Tom Sawyer 1930 Not checked out
667 Midway 2019 Not checked out
12 Fear 1990 Checked out by User 1
9 Bird Box 2018 Checked out by User 233
```

(b) output.txt

Figure 3: Commands file and output file according to commands.txt

4 Grading and Evaluation

- Your work will be graded over a maximum of 100 points.
- Your total score will be partial according to the grading policy stated below.

Reading the file contents and building the linkedlist	15p
Adding movie and user	10p
Deleting movie and user	10p
Checking out a movie and returning a movie	15p
Returning a movie	10p
Showing the list of all movies	10p
Showing detailed information about a particular movie and user	15p
Code design, clean and readable code, algorithmic perspective, comments	15p

- Your code will be tested with different inputs. And one output file will be expected as outputs file.

Usage example:
>g++ -o Main Main.cpp
>./Main commands.txt output.txt

Notes

- Do not miss the deadline.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- **For this assignment, you must use your own implementation of dynamic 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). And also you should not use array-based list for this assignment.**
- Write READABLE SOURCE CODE block
- You can ask your questions via Piazza (https://piazza.com/hacettepe.edu.tr/fall/bbm201) and you are supposed to be aware of everything discussed in Piazza.
- You will use online submission system to submit your experiments. https://submit.cs.hacettepe.edu.tr/ No other submission method (email or etc.) will be accepted. Do not submit any file via e-mail related with this assignment.
- File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system). You must submit your work with the file hierarchy stated below:

<student_id.zip>/(Required)
→src/(Required)
→Main.cpp(Required)
→LibrarySystem.cpp(Required)
→LibrarySystem.h(Required)
→*.cpp(optional)
→*.h(optional)

Policy

All work on assignments must be done **individually** unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an **abstract** way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) **will not be tolerated**. In short, turning in someone else's work (from internet), in whole or in part, as your own will be considered **as a violation of academic integrity**. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.