CS 201, Fall 2024

Homework Assignment 3

Due: 23:59, December 13, 2024

1 Introduction

In this homework, you will create a music streaming service called BilkentBeats, implemented using linked lists. The music streaming service consists of users, songs, and playlists, each identified by unique IDs across the system. Note that these ID types are independent of each other. For instance, there cannot be more than one user with ID 6, but there can be a playlist with ID 6 and a song with ID 6. They would be different entities.

In this system, you will store a linked list of users and a linked list of songs. For each user, you will store a linked list of playlists. For each playlist, you will store a linked list of songs IDs. In your implementation, you MUST use **sorted linked lists** for storing the users, songs, and playlists, and MUST use **unsorted linked lists** for storing the songs in playlists.

1.1 Users

Each user has a unique ID and a name. You need to use a sorted linked list to store the items with respect to ascending user IDs. The required functions are as follows:

- `addUser`: Adds a new user to the system with an ID and a name. Since IDs must be unique, the system must check whether or not the specified user ID already exists, and if it exists, it must not allow the operation and display a warning message. Initially, a user does not have any playlist. Example log messages:
  - Added user 6.
  - Cannot add user. There is already a user with ID 6.
- `removeUser`: The system will allow removing users. If you remove the user, all related playlists are removed as well. If the user does not exist, the system must display a warning message. Example log messages:
  - Removed user 6.
  - Cannot remove user. There is no user with ID 6.
- `printUsers`: Prints IDs, user names, and also list of playlist IDs. If there is no user in the system, give a warning message. The entries must be shown in ASCENDING ORDER with respect to the IDs. Example log messages:

  - There are no users to show.
  - Users in the system:
    User ID : 21, Name : Bora, Playlist IDs : [16,18]
    User ID : 25, Name : Filiz, Playlist IDs : None
    User ID : 26, Name : Ebru, Playlist IDs : [19]
    User ID : 29, Name : Melda, Playlist IDs : [15]

1.2 The Music Library

BilkentBeats stores all songs in the music library as a sorted linked list in ascending order by song ID. You need to use this linked list to access the information about the songs. Each song is identified by a unique ID. Songs also store their names and the names of the artists. Note that a song can be in multiple playlists. The required functions are as follows:

- `addSong`: Adds a song to the BilkentBeats music library. If the song was already added, give a warning message and do not allow the operation. Example log messages:
  - Added song 23.
  - Cannot add song. BilkentBeats already contains song 23.
- `removeSong`: Removes the song from the BilkentBeats music library. If the song does not exist, give a warning message. If the song is successfully removed, you need to ensure all user playlists are updated accordingly. Example log messages:
  - Removed song 23.
  - Cannot remove song. There is no song with ID 23.
- `printSongs`: Prints the IDs, names, and artists of all songs in the music library. If there is no song in the music library, give a warning message. The entries should be shown in ASCENDING ORDER according to IDs. Example log messages:
  - Cannot print songs. There is no song in the music library.
  - Music Library:
    Song 23 : Bohemian Rhapsody - Queen
    Song 27 : Billie Jean - Michael Jackson
    Song 46 : Nature Boy - Nat King Cole
    Song 89 : Starman - David Bowie
    Song 92 : Imagine - John Lennon

1.3 Playlists

Each playlist is identified by a unique ID. It also has a list of song IDs. Each playlist can belong to a single user. Note that the songs in a playlist are not sorted according to their song IDs but

they appear in the order in which they are added to the playlist. The required functions are as follows:

- `addPlaylist`: Adds a new playlist to the user with an ID. The system must first check whether user ID exists. If it exists, then it should check whether specified playlist ID exists in any user. If the specified user does not exist or there is already a playlist with the same ID in a user, it must not allow the operation and display a warning message. Initially, a playlist does not contain any songs. Example log messages:
  - Added playlist 6 to user 4.
  - Cannot add playlist. There is no user with ID 4.
  - Cannot add playlist. There is a user having a playlist with ID 6.
- `removePlaylist`: The system will allow removing playlists from users. If the user does not exist or the user does not contain the specified playlist, the system must display a warning message. Example log messages:
  - Removed playlist 6 from user 4.
  - Cannot remove playlist. There is no user with ID 4.
  - Cannot remove playlist. User 4 does not have a playlist with ID 6.
- `addSongToPlaylist`: Adds a song to the playlist. If the specified playlist ID does not exist, or the song does not exist in the music library, or the song is already in the playlist, give a warning message and do not allow the operation. Example log messages:
  - Added song 23 to playlist 6.
  - Cannot add song. There is no playlist with ID 6.
  - Cannot add song. There is no song with ID 23.
  - Cannot add song. The playlist already contains song 23.
- `removeSongFromPlaylist`: Removes the song from the specified playlist. If the playlist does not exist or the playlist does not contain the specified song, give a warning message. Example log messages:
  - Removed song 23 from playlist 6.
  - Cannot remove song. There is no playlist with ID 6.
  - Cannot remove song. There is no song 23 in playlist 6.
- `printSongsInPlaylist`: Prints the IDs, names, and artists of the songs for the given playlist. If the specified playlist does not exist, give a warning message. The entries should be displayed in order of adding to the playlist. Example log messages:
  - Cannot print songs. There is no playlist with ID 4.

– There are no songs to show in playlist 4.

– Songs in playlist 4:
  Song 23 : Bohemian Rhapsody - Queen
  Song 89 : Starman - David Bowie
  Song 46 : Nature Boy - Nat King Cole

Below is the required public part of the `BilkentBeats` class that you must write in this assignment. The name of the class must be `BilkentBeats`, and must include these public member functions. The interface for the class must be written in the file called `BilkentBeats.h` and its implementation must be written in the file called `BilkentBeats.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 and implement them in separate files.

```
1  class BilkentBeats {
2  public:
3      BilkentBeats();
4      ~BilkentBeats();
5
6      void addUser( const int userId, const string userName );
7      void removeUser( const int userId );
8      void printUsers() const;
9
10     void addSong( const int songID, const string songName, const string artists );
11     void removeSong( const int songID );
12     void printSongs() const;
13
14     void addPlaylist( const int userId, const int playlistId );
15     void removePlaylist( const int userId, const int playlistId );
16     void addSongToPlaylist( const int playlistId, const int songId );
17     void removeSongFromPlaylist( const int playlistId, const int songId );
18     void printSongsInPlaylist( const int playlistId ) const;
19
20 };
```

Here is an example test program that uses this class and the corresponding output. We will use a similar program to test your solution so make sure that the name of the class is `BilkentBeats`, its interface is in the file called `BilkentBeats.h`, and the required functions are defined as shown above. Your implementation **MUST** use exactly the same format given in the example output to display the messages expected as the result of the defined functions.

Example test code:

```
int main() {

    BilkentBeats bilkentBeats;

    bilkentBeats.addSong(45, "Stairway to Heaven","Led Zeppelin");
    bilkentBeats.addSong(42, "Billie Jean", "Michael Jackson");
    bilkentBeats.addSong(43, "Hotel California", "Eagles");
    bilkentBeats.addSong(48, "Imagine", "John Lennon");
    bilkentBeats.addSong(42, "Bohemian Rhapsody", "Queen");
    cout<<endl;

    bilkentBeats.printSongs();
    cout<<endl;

    bilkentBeats.addUser(26, "Ebru");
    bilkentBeats.addUser(21, "Bora");
    bilkentBeats.addUser(29, "Melda");
    bilkentBeats.addUser(25, "Filiz");
    bilkentBeats.addUser(21, "Bugra");
    cout<<endl;

    bilkentBeats.addPlaylist(21,16);
    bilkentBeats.addPlaylist(21,18);
    bilkentBeats.addPlaylist(26,19);
    bilkentBeats.addPlaylist(29,15);
    bilkentBeats.addPlaylist(26,16);
    bilkentBeats.addPlaylist(49,13);
    cout<<endl;

    bilkentBeats.printUsers();
    cout<<endl;

    bilkentBeats.addSongToPlaylist(16,45);
    bilkentBeats.addSongToPlaylist(19,45);
    bilkentBeats.addSongToPlaylist(19,43);
    bilkentBeats.addSongToPlaylist(18,42);
    bilkentBeats.addSongToPlaylist(16,45);
    bilkentBeats.addSongToPlaylist(86,43);
    bilkentBeats.addSongToPlaylist(16,86);
    cout<<endl;

    bilkentBeats.printSongsInPlaylist(16);
    cout<<endl;
    bilkentBeats.printSongsInPlaylist(19);
```

```
45      cout<<endl;
46      bilkentBeats.printSongsInPlaylist(18);
47      cout<<endl;
48      bilkentBeats.printSongsInPlaylist(65);
49      cout<<endl;
50
51      bilkentBeats.removeSong(48);
52      bilkentBeats.removeSong(45);
53      bilkentBeats.removeSong(95);
54      cout<<endl;
55
56      bilkentBeats.printSongsInPlaylist(19);
57      cout<<endl;
58      bilkentBeats.printSongsInPlaylist(18);
59      cout<<endl;
60      bilkentBeats.printSongsInPlaylist(16);
61      cout<<endl;
62
63      bilkentBeats.removePlaylist(26,19);
64      bilkentBeats.removePlaylist(26,15);
65      bilkentBeats.removePlaylist(92,16);
66      cout<<endl;
67
68      bilkentBeats.removeSongFromPlaylist(18,42);
69      bilkentBeats.removeSongFromPlaylist(19,43);
70      bilkentBeats.removeSongFromPlaylist(18,85);
71      cout<<endl;
72
73      bilkentBeats.removeUser(21);
74      bilkentBeats.removeUser(25);
75      bilkentBeats.removeUser(95);
76      cout<<endl;
77
78      bilkentBeats.printUsers();
79      cout<<endl;
80
81      bilkentBeats.printSongs();
82      cout<<endl;
83
84      bilkentBeats.removeUser(26);
85      bilkentBeats.removeUser(29);
86      bilkentBeats.removeSong(42);
87      bilkentBeats.removeSong(43);
88      cout<<endl;
89
90      bilkentBeats.printUsers();
```

```
    cout<<endl;

    bilkentBeats.printSongs();

    return 0;
}
```

Output of the example test code:

```
Added song 45.
Added song 42.
Added song 43.
Added song 48.
Cannot add song. BilkentBeats already contains song 42.

Music Library:
Song 42 : Billie Jean - Michael Jackson
Song 43 : Hotel California - Eagles
Song 45 : Stairway to Heaven - Led Zeppelin
Song 48 : Imagine - John Lennon

Added user 26.
Added user 21.
Added user 29.
Added user 25.
Cannot add user. There is already a user with ID 21.

Added playlist 16 to user 21.
Added playlist 18 to user 21.
Added playlist 19 to user 26.
Added playlist 15 to user 29.
Cannot add playlist. There is a user having a playlist with ID 16.
Cannot add playlist. There is no user with ID 49.

Users in the system:
User ID : 21, Name : Bora, Playlist IDs : [16,18]
User ID : 25, Name : Filiz, Playlist IDs : None
User ID : 26, Name : Ebru, Playlist IDs : [19]
User ID : 29, Name : Melda, Playlist IDs : [15]

Added song 45 to playlist 16.
Added song 45 to playlist 19.
Added song 43 to playlist 19.
Added song 42 to playlist 18.
Cannot add song. The playlist already contains song 45.
Cannot add song. There is no playlist with ID 86.
```

```
Cannot add song. There is no song with ID 86.

Songs in playlist 16:
Song 45 : Stairway to Heaven - Led Zeppelin

Songs in playlist 19:
Song 45 : Stairway to Heaven - Led Zeppelin
Song 43 : Hotel California - Eagles

Songs in playlist 18:
Song 42 : Billie Jean - Michael Jackson

Cannot print songs. There is no playlist with ID 65.

Removed song 48.
Removed song 45.
Cannot remove song. There is no song with ID 95.

Songs in playlist 19:
Song 43 : Hotel California - Eagles

Songs in playlist 18:
Song 42 : Billie Jean - Michael Jackson

There are no songs to show in playlist 16.

Removed playlist 19 from user 26.
Cannot remove playlist. User 26 does not have a playlist with ID 15.
Cannot remove playlist. There is no user with ID 92.

Removed song 42 from playlist 18.
Cannot remove song. There is no playlist with ID 19.
Cannot remove song. There is no song 85 in playlist 18.

Removed user 21.
Removed user 25.
Cannot remove user. There is no user with ID 95.

Users in the system:
User ID : 26, Name : Ebru, Playlist IDs : None
User ID : 29, Name : Melda, Playlist IDs : [15]

Music Library:
Song 42 : Billie Jean - Michael Jackson
Song 43 : Hotel California - Eagles

```

```
Removed user 26.
Removed user 29.
Removed song 42.
Removed song 43.

There are no users to show.

Cannot print songs. There is no song in the music library.
```

2 Specifications

1. You ARE NOT ALLOWED to modify the given parts of the header file. You MUST use **linked lists** in your implementation (sorted or unsorted, according to the requirements described earlier). You will get no points if you use dynamic or fixed-sized arrays or any other data structures such as vectors/arrays/lists from the standard library. However, if necessary, you may define additional data members and member functions. You may also define additional classes.

2. Moreover, you ARE NOT ALLOWED to use any global variables or any global functions.

3. Output message for each operation MUST match the format shown in the output of the example code. Your output will be compared verbatim with the expected output during evaluation.

4. 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. To detect memory leaks, you may want to use Valgrind which is available at http://valgrind.org.

3 Submission

1. In this assignment, you must have separate interface and implementation files (i.e., separate `.h` and `.cpp` files) for your class. Your class name MUST BE `BilkentBeats` and your file names MUST BE `BilkentBeats.h` and `BilkentBeats.cpp`. Note that you may write additional class(es) in your solution.

2. The code (`main` function) given above is just an example. We will test your implementation using different scenarios, which will contain different function calls. Thus, do not test your implementation only by using this example code. We recommend you to write your own driver files to make extra tests. However, you MUST NOT submit these test codes (we will use our own test code). In other words, do not submit a file that contains a function called `main`.

3. You should put all of your `.h` and `.cpp` files into a folder and zip the folder (in this zip file, there should not be any file containing a `main` function). The name of this zip file should

conform to 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. Make sure that each file that you submit (each and every file in the archive) contains your name, section, and student number at the top as comments.

5. You are free to write your programs in any environment (you may use Linux, Windows, MacOS, etc.). 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. Your code will be tested by using an automated test suite that includes multiple test cases where each case corresponds to a specific number of points in the overall grade. We will provide you with example test cases by email. Thus, we strongly recommend you to make sure that your program successfully compiles and correctly works on dijkstra.ug.bcc.bilkent.edu.tr before submitting your assignment. If your current code does not fully compile on dijkstra before submission, you can try to comment out the faulty parts so that the remaining code can be compiled and tested during evaluation.

6. This assignment is due by 23:59 on Friday, December 13, 2024. You should upload your work to Moodle before the deadline. No hardcopy submission is needed. Late submissions will not be accepted (if you can upload to Moodle, then you are fine). There will be no extension to this deadline.

7. We use an automated tool as well as manual inspection to check your submissions against plagiarism. For questions regarding academic integrity and use of external tools (including generative AI tools), please refer to the course home page and the Honor Code for Introductory Programming Courses (CS 101/102/201/202) at `https://docs.google.com/document/d/1v_3ltpV_lClLsROXrMbojyuv4KrFQAm1uoz3SdC-7es/edit?usp=sharing`.

8. This homework will be graded by your TA **Mehmet Alper Yilmaz** (mehmet.yilmaz@bilkent.edu.tr). Thus, you may ask your homework related questions directly to him. There will also be a forum on Moodle for questions.