CS 201, Fall 2022
Homework Assignment 1

Due: 23:59, November 6, 2022

In this homework, you will implement a sports league management system. The league consists of teams with unique names. Each team also has a set of players, named the roster of the team. The management system should support adding and removing teams to the league and adding and removing players to the teams. In your implementation, you MUST use dynamically allocated arrays for storing the team and player information.

The league management system will have the following functionalities. The details of these functionalities are given below:

1. Add a team to the league
2. Remove a team from the league
3. Add a player to a team
4. Remove a player from a team
5. Transfer a player from one team to another team
6. Show the list of all teams
7. Show detailed information about a team
8. Show detailed information about a player

Add a team to the league: The management system will allow the user to add a team indicating its name and the year it was founded. Since the team names must be unique, the system should check whether or not the specified team already exists, and if it exists, it should not allow the operation and display a warning message. Initially a team does not have any players.

Remove a team from the league: The management system will allow the user to remove an existing team indicated by its name. If the team does not exist, the system should display a warning message. Note that this operation will also remove all players from this team.

Add a player to a team: The management system will allow to add a new player to a particular team. Each player has a name, a jersey number, and a salary amount. The system should first check whether or not the team exists. If it does not, it should prevent to add the player and display a warning message. The system should also check whether or not a player with this jersey number already exists in the team. If it does, it should prevent to add the player and display a warning message. Note that the jersey numbers must be unique within a team.

Remove a player from a team: The management system will allow to remove an existing player from a team. If the team does not exist or the player does not exist in the team, the system should display a warning message.

Transfer a player from one team to another team: The management system will allow to transfer a player between two teams. If at least one of the teams does not exist or the player does not exist in the departing team, the system should display a warning message. If another player with the same jersey number exists in the arrival team, the system should prevent this transfer and display a warning message.

Show the list of all teams: The management system will allow to display a list of all teams. The list includes the team name, the year it was founded, the number of players in the roster, and the total salary for these players.

Show detailed information about a team: The management system will allow to specify the name of a team and display detailed information about that particular team. This information includes the name of the team, its founding year, information about the players in its roster, and the total salary for the players. If the team does not exist, the system should display a warning message.

Show detailed information about a player: The management system will allow to specify the name of a player and display detailed information about that particular player. This information includes the name of the player, the jersey number, salary, and the team the player is currently playing in. If the player does not exist, the system should display a warning message.

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

```
class LeagueManagementSystem {

public:
    LeagueManagementSystem();
    ~LeagueManagementSystem();

    void addTeam( const string name, const int year );
    void removeTeam( const string name );
    void addPlayer( const string teamName, const string playerName,
                    const int jersey, const int salary );
    void removePlayer( const string teamName, const string playerName );
    void transferPlayer( const string playerName,
                         const string departTeamName, const string arriveTeamName );

    void showAllTeams() const;
    void showTeam( const string name ) const;
    void showPlayer( const string name ) const;
};
```

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 `LeagueManagementSystem`, its interface is in the file called `LeagueManagementSystem.h`, and the required functions are defined as shown above. Your implementation should use the format given in the example output to display the messages expected as the result of the defined functions.

Example test code:

```
#include <iostream>
using namespace std;

#include "LeagueManagementSystem.h"

int main() {

    LeagueManagementSystem LMS;

    LMS.showAllTeams();
    cout << endl;
```

```
    LMS.addTeam( "Fenerbahce", 1907 );
    LMS.addTeam( "Tofas", 1957 );
    LMS.addTeam( "Anadolu Efes", 1951 );
    LMS.removeTeam( "Tofas" );
    LMS.addTeam( "Darussafaka", 1951 );
    LMS.addTeam( "Fenerbahce", 1907 );
    LMS.removeTeam( "Tofas" );
    cout << endl;

    LMS.addPlayer( "Fenerbahce", "Melih Mahmutoglu", 10, 2000000 );
    LMS.addPlayer( "Fenerbahce", "Sehmus Hazer", 2, 1000000 );
    LMS.addPlayer( "Darussafaka", "Ercan Osmani", 6, 1000000 );
    LMS.addPlayer( "Anadolu Efes", "Dogus Balbay", 4, 1000000 );
    LMS.addPlayer( "Anadolu Efes", "Bugrahan Tuncer", 10, 1500000 );
    LMS.removePlayer( "Darussafaka", "Ercan Osmani" );
    cout << endl;

    LMS.addPlayer( "Fenerbahce", "Stephen Curry", 10, 10000000 );
    LMS.addPlayer( "Karsiyaka", "Metecan Birsen", 1, 1000000 );
    LMS.addTeam( "Karsiyaka", 1912 );
    LMS.addPlayer( "Karsiyaka", "Metecan Birsen", 1, 1000000 );
    LMS.removePlayer( "Tofas", "Melih Mahmutoglu" );
    LMS.removePlayer( "Fenerbahce", "Bugrahan Tuncer" );
    LMS.transferPlayer( "Metecan Birsen", "Karsiyaka", "Fenerbahce" );
    LMS.transferPlayer( "Lebron James", "Los Angeles Lakers", "Fenerbahce" );
    LMS.transferPlayer( "Michael Jordan", "Darussafaka", "Fenerbahce" );
    LMS.transferPlayer( "Bugrahan Tuncer", "Anadolu Efes", "Fenerbahce" );
    cout << endl;

    LMS.showAllTeams();
    cout << endl;
    LMS.showTeam( "Fenerbahce" );
    cout << endl;
    LMS.showTeam( "Boston Celtics" );
    cout << endl;
    LMS.showPlayer( "Melih Mahmutoglu" );
    cout << endl;
    LMS.showPlayer( "Furkan Korkmaz" );

    return 0;
}
```

Output of the example test code:

```
Teams in the league management system:
None

Added team Fenerbahce.
Added team Tofas.
Added team Anadolu Efes.
Removed team Tofas.
Added team Darussafaka.
Cannot add team. Team Fenerbahce already exists.
Cannot remove team. Team Tofas does not exist.
```

```
Added player Melih Mahmutoglu to team Fenerbahce.
Added player Sehmus Hazer to team Fenerbahce.
Added player Ercan Osmani to team Darussafaka.
Added player Dogus Balbay to team Anadolu Efes.
Added player Bugrahan Tuncer to team Anadolu Efes.
Removed player Ercan Osmani from team Darussafaka.

Cannot add player. Jersey number 10 already exists in team Fenerbahce.
Cannot add player. Team Karsiyaka does not exist.
Added team Karsiyaka.
Added player Metecan Birsen to team Karsiyaka.
Cannot remove player. Team Tofas does not exist.
Cannot remove player. Player Bugrahan Tuncer does not exist.
Transferred player Metecan Birsen from team Karsiyaka to team Fenerbahce.
Cannot transfer player. Team Los Angeles Lakers does not exist.
Cannot transfer player. Player Michael Jordan does not exist.
Cannot transfer player. Jersey number 10 already exists in team Fenerbahce.

Teams in the league management system:
Fenerbahce, 1907, 3 players, 4000000 TL total salary
Anadolu Efes, 1951, 2 players, 2500000 TL total salary
Darussafaka, 1951, 0 players, 0 TL total salary
Karsiyaka, 1912, 0 players, 0 TL total salary

Team:
Fenerbahce, 1907, 3 players, 4000000 TL total salary
Players:
Melih Mahmutoglu, jersey 10, 2000000 TL salary
Sehmus Hazer, jersey 2, 1000000 TL salary
Metecan Birsen, jersey 1, 1000000 TL salary

Team Boston Celtics does not exist.

Player:
Melih Mahmutoglu, jersey 10, 2000000 TL salary
Plays in team Fenerbahce.

Player Furkan Korkmaz does not exist.
```

IMPORTANT NOTES:
Do not start your homework before reading these notes!!!

NOTES ABOUT IMPLEMENTATION:

1. You ARE NOT ALLOWED to modify the given parts of the header file. You MUST use dynamically allocated arrays with only the necessary amount of memory in your implementation. That is, if there are 10 teams in the system, it should use memory only for these 10 teams. In other words, you cannot initially allocate a large array for teams and expect it to get filled later. The same argument applies to space used to store players. You will get no points if you use fixed-sized arrays, linked lists or any other data structures such as vectors/arrays from the standard library. However, if necessary, you may define additional data members and member functions.
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.

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`.

5. Otherwise stated in the description, you may assume that the inputs for the functions are always valid (e.g., foundation year of a team is a valid positive integer number) so that you do not need to make any input checks.

6. You can also assume that player names will be entered as unique names in the system. Thus, you do not need to make additional checks for the uniqueness of player names.

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. Your class name MUST BE `LeagueManagementSystem` and your file names MUST BE `LeagueManagementSystem.h` and `LeagueManagementSystem.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. If we could not get your program properly work on the dijkstra machine, you would lose a considerable amount of points. Thus, we recommend you to make sure that your program compiles and properly works on dijkstra.ug.bcc.bilkent.edu.tr before submitting your assignment.

6. This assignment is due by 23:59 on Sunday, November 6, 2022. You should upload your work to Moodle before the deadline. No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the course home page for further discussion of the late homework policy.

7. We use an automated tool as well as manual inspection to check your submissions against plagiarism. Please see the course home page for further discussion of academic integrity and the honor code for programming courses in our department.

8. This homework will be graded by your TA Cihan Erkan (cihan.erkan at bilkent.edu.tr). Thus, you may ask your homework related questions directly to him. There will also be a forum on Moodle for questions.