CSCI804 Assignment 2

Due: by September 14, 15:00. **Resubmission is open**: till September 22, 15:00
Marks: 15 marks

Objective: Get practical experience in using
- C++ classes, constructors, destructors, operators
- relationship between classes
- UML diagrams

General Requirements

- Follow the principles of OO programming when you design your classes
- Put your name, student number at the beginning of each file submitted

```
/*------------------------------------------------------
Student's Name:
Student's email address:
Laboratory group (group code and time):
Purpose of this assignment:
-------------------------------------------------------*/
```

- Add comments to the source code to make your solution easier to follow

Assignment requirements:

Write a program that simulates a championship (soccer, handball, hockey, etc).

BACKGROUND

The structure of the simulation is such that a given number of teams play in a league on a double round-robin basis, in which every team plays with all others in the league once at home and once away. It means that if there are n teams in a league, there would be $n*(n-1)$ games played.

Teams receive thwo points for a win and one point for a draw. No points are awarded for a loss.

Teams are ranked according to the following criteria:

1. By the total points first;
2. For teams with an equal number of points, by goal difference (i.e. the number of goals scored for minus the number of goals scored against);
3. If the goal difference is also the same, by goals scored for.
4. If still equal, the teams will occupy the same position.

CODING REQUIREMENTS

- Your program shall be well structured and designed using the concept of **object oriented** programming:
  - follow the fundamental principles of the OOM when you define a class
  - Your program shall be subdivided into three files `classes.h` `classes.cpp` and `a2main.cpp`

The `main()` function of the program shall read two arguments from its command line:

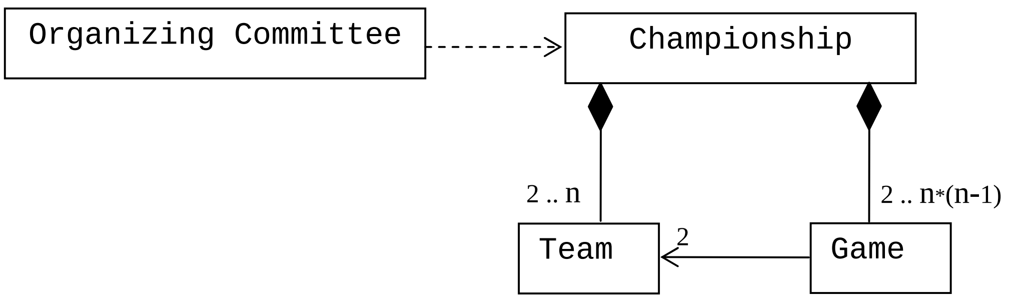
1. The league type: a c-string that may be equal only to `soccer`, `handball`, or `hockey`
2. The number of teams: an integer value that is greater than 1, as the minimum number of teams for playing a game is 2.

If the provided values of the league type or the number of teams do not satisfy the requirements, the program shall output an error message into the standard error stream and terminate further execution. If the arguments are correct, the `main()` function shall create an instance of a class `Championship` passing two parameters to its constructor -the league type and the number of teams. The start of the championship has to be initialised by the Organizing Committee

For example:

```
OrganizingComittee uaeh;
Championship *eu2016 = new Championship( "hockey", 5 );
uaeh.start( eu2016 );
```

The program shall have the structure as shown below:



A UML class diagram that describes the program structure

The class `Championship` shall have a c-string (or a string) type data member to store the league type. It also shall create the required number of objects `Team`. It shall generate all possible *n**(*n* – 1) games played in a random order among the teams for the whole season. It shall run the championship when a member function `Championship::runChampionship()` is called from `OrganizingComittee::start(…)`. Once all games have been played, one of the member functions shall print a table with the results of the championship according to the required output format (see below).

The class `Team` shall have private data members to store a team name, and also total points, goals scored for and goals scored against which should be updated during the championship. The private data members points, goals scored for and goals scored against shall be updated and accessed through public methods. To make implementation simple, team names should be: Team1, Team2, Team3,… You also need to implement the `< , >, ==, !=` overloaded operators to compare teams based on the ranking criteria described earlier. If other operators are needed, they should be implemented too.

The class `Game` shall be associated with two teams - the home team and the visiting team. The class shall have data members to store the number of goals scored by the home team, and the number of goals scored by the visitors in a single game. This class shall have a method to simulate a game by generating randomly the number of goals scored by each team, thus simulating a random outcome for each game. The maximum number of goals which a team can score in a single game must be limited to 15.

The objects of type `Team` and `Game` shall be created using dynamic memory allocation.
Before you start writing C++ code, consider what data members and member functions are needed for each class. Also consider how to implement the specified relationship between classes.

The program shall properly manage dynamic memory allocation.
You can use the command `bcheck` on banshee to check if there is any memory leak.

DEVELOPMENT and TESTING

You need to follow good programming practices when you write the source code:
- meaningful identifiers for data members, member functions, classes
- no global variables
- appropriate indentation
- appropriate comments

To minimize time needed to complete this assignment, develop your program incrementally testing each new component added. Make sure you program is robust and does not crash. The program shall not produce memory leaks.

As assignments are intended to assess students' understanding of the subject material covered, each assignment must be solved using only material covered up to that point in the lectures.

Target platform: Sun UNIX (**banshee**), g++

Upload all source code files to your working directory on `banshee`.

Compile the program in the working directory on `banshee` using g++ compiler

```
g++ –o assignment2.exe –Wall –ansi a2main.cpp classes.cpp
```

Run the program with two arguments provided:

```
assignment2.exe soccer 3
```

Your program should produce output in the following format:

```
*** Soccer League Championship ***
- Participant list -
Team1
Team2
Team3

- Games played -
Team1 Team2 0 0
Team3 Team2 2 2
Team2 Team1 2 4
Team3 Team1 1 1
Team2 Team3 3 3
Team1 Team3 2 1

- Championship table -
Position | Team | Points | GoalDiff | GoalsFor | GoalsAgainst
   1      Team1    6          3         7            4
   2      Team3    3         -1         7            8
   3      Team2    3         -2         7            9
```

This example shows only the required output format. The actual numbers will be different. Prepare several test cases to test your program properly.

Submission:

All assignments must be submitted electronically via the `submit` system. For this assignment you must submit three files via the command (**in one line**):

```
submit -u your_user_name -c CSCI204 -a 2 a2main.cpp classes.h classes.cpp
```

and enter your password.

Make sure that you use the **correct file names. The UNIX system is case sensitive.** You must submit all files **in one *submit* command line.**

- Files with other names will not be tested and marked.
- Submission via e-mail is NOT acceptable.

After submit your assignment successfully, please check your email of confirmation. You should keep the email for the reference.

NOTES:
1. Submit your assignment before the due date. You can resubmit later if necessary. Only the latest submission will be tested and marked.
2. SUBMISSION BY EMAIL IS NOT ACCEPTABLE. YOU HAVE TO USE SUBMIT COMMAND TO SUBMIT YOUR WORK.
3. ASSIGNMENT FILES WITHOUT PROPERLY FILLED HEADERS WILL NOT BE MARKED
4. DO NOT FORWARD THE ACKNOWLEDGEMENT FROM THE SERVER THAT YOU WILL RECEIVE AFTER SUBMITTING YOUR ASSIGNMENT TO ANY OTHER EMAIL ACCOUNT, OTHER THAN UNIVERSITY'S EMAIL ACCOUNT.
THIS ACKNOWLEDGEMENT MAY BE NEEDED IF YOU WANT TO QUERY ABOUT YOUR ASSIGNMENT.
WITHOUT THE ACKNOWLEDGEMENT FROM THE SERVER, YOUR QUERY WILL NOT BE PROCESSED
5. Enquiries about the marks can only be made within a maximum of 1 week after the assignment results are published. After 1 week the marks cannot be changed.