

CSCI204 Assignment 2

Due: by April 16, 14:30. Resubmission is open: till April 23, 14:30

Marks: 16 marks

Objective: get practical experience in using:

- C++ classes, constructors, destructors, operators
 - relationship between classes
 - UML diagrams
-

General Requirements

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

```
/*-----  
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 three 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.

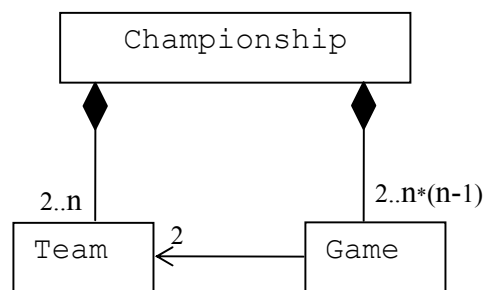
Implementation requirements

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

1. The league type: a string that may be equal 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 program shall have the structure as shown below:



A UML 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 then. 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 10.

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 to avoid memory leaks. You can use the command `bcheck` on banshee to check if there is any memory leak.

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

Testing:

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

Compile the program in the working directory on banshee using CC compiler

```
CC -o assignment2.exe 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  8         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 all the 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**.

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

NOTES:

1. SUBMIT AS EARLY AS POSSIBLE. YOU CAN RESUBMIT LATER IF NECESSARY. Only the latest submission will be marked.
3. SUBMISSION VIA EMAIL IS NOT ACCEPTABLE. YOU HAVE TO USE SUBMIT COMMAND TO SUBMIT YOUR WORK.
4. ASSIGNMENT WITHOUT PROPERLY FILLED ASSIGNMENT HEADERS IN SUBMITTED FILES WILL NOT BE MARKED
4. The submitted file names must be the same as in the submission example. Files with other names will not be tested by the submit system and therefore will not be marked.
5. Enquiries about the marks can only be made within a maximum of 1 week after the assignment results are published. After one week the marks cannot be changed.
6. The assignment is an **individual assignment** and it is expected that all its tasks will be solved **individually without any cooperation** with the other students. If you have any doubts, questions, etc. please consult your lecturer. Plagiarism will result in a **FAIL** grade being recorded for that assessment task.