CMPE 260 - Project 1

Due: 23 April 2018

This project will test your knowledge of logic programming and Prolog. Your project will be graded on correctness, readability, testing strategy, efficiency, documentation, and the use of logic programming style. Your task is to write Prolog predicates (statements) for manipulating lists of football teams, and the matches played between them. You may use additional predicates as needed in defining these predicates.

Problem Description

Copy the file **cl_base.pl** posted on the class website into your working directory. The file contains a database of facts about 4 teams and the played matches that you may use to test and debug your program. The predicates that will be used are as follows:

```
team(teamName, hometown).
match(week, homeTeam, homeTeamScore, awayTeam, awayTeamScore).
```

The following is a portion of a sample database illustrating the two relations that are defined:

```
team(realmadrid, madrid).
team(juventus, torinp).
team(galatasaray, istanbul).
team(kobenhavn, kopenag).

match(1, galatasaray, 1, realmadrid, 6).
match(1, kobenhavn, 1, juventus, 1).

match(2, juventus, 2, galatasaray, 2).
match(2, realmadrid, 4, kobenhavn, 0).
```

The first predicate shows the teams currently existing in the database. For example `team(galatasaray, istanbul).` means that galatasaray is a team that plays home matches in istanbul.

The second predicate implies that there has been a match between the two given teams in the given week. Score is also given in the predicate. To illustrate, `match(2, realmadrid, 4, kobenhavn, 0).` implies that "In the 2nd week, realmadrid defeated kobenhavn with the score of 4-0.

You will implement the predicates described below, as well as any necessary additional predicates to support them. You will need to load the database file into Prolog. When you are testing, you can simply copy and paste the facts in your working file but you must put the predicates that you implement into a separate file called **predicates.pl** when submitting.

Your Tasks

0.1 All Teams

Implement the predicate `allTeams(L,N)`. L is the list containing all the teams in the database where N is the number of elements in the list.

In Prolog you can continue querying by pressing `;` and the next result is retrieved. If there is no other result left, then it returns **False**. If you don't make a new query and press enter it returns **True**.

In this predicate, you should be able to specify queries **with any combination of variables and constants** as shown in the first query below. In the first query, I pressed ; two times and then I pressed enter. In the second and the third query, I checked if the given data is correct according to knowledge base. In the fourth query I asked to show list of teams and in the fifth query I asked to show number of teams.

```
?- allTeams(L,N).
L = [galatasaray, realmadrid, juventus, kobenhavn, manutd, realsociedad, shaktard, bleverkusen,
omarseille, arsenal, fcnapoli, bdortmund]
N = 12;
L = [galatasaray, realmadrid, juventus, kobenhavn, manutd, realsociedad, shaktard, bleverkusen,
omarseille, arsenal, bdortmund, fcnapoli]
N = 12;
L = [galatasaray, realmadrid, juventus, kobenhavn, manutd, realsociedad, shaktard, bleverkusen,
omarseille, bdortmund, arsenal, fcnapoli]
N = 12
True

?- allteams([galatasaray, realmadrid, juventus, kobenhavn, manutd, realsociedad, shaktard,
bleverkusen, omarseille, arsenal, fcnapoli, bdortmund], 12).
True

?- allteams([], 12).
False

?- allteams(L, 12).
L = [galatasaray, realmadrid, juventus, kobenhavn, manutd, realsociedad, shaktard,
bleverkusen, omarseille, bdortmund, arsenal, fcnapoli]
True

?- allteams([galatasaray, realmadrid, juventus, kobenhavn, manutd, realsociedad, shaktard,
bleverkusen, omarseille, bdortmund, arsenal, fcnapoli], N).
N = 12
False
```

0.2 Match Results

Implement the following predicates:

- `wins(T,W,L,N)` implies that L involves the teams defeated by team T when we are in week W and N is the number of elements in L.
- `losses(T,W,L,N)` implies that L involves the teams that defeated team T when we are in week W and N is the number of elements in L.
- `draws(T,W,L,N)` is very similar but now L involves the teams that team T could not defeat also did not lose to.

You can assume that T and W will be given as constants. Note that when we say, we are in week W, that means there have been W matches played by a team. So you will consider all these matches played by the queried team.

```
?- wins(galatasaray,4,L,N).
L = [kobenhavn]
N = 1 ;
False

?- losses(galatasaray,4,L,N).
```

```
L = [realmadrid, kobenhavn]
N = 2 ;
False

?- draws(galatasaray,4,L,N).
L = [juventus]
N = 1 ;
False

?- draws(galatasaray,4,[juventus],N).
N = 1 ;
False

?- draws(galatasaray,4,L,1).
L = [juventus] ;
False

?- draws(galatasaray,4,[juventus],1).
True
```

0.3 Goals Socred and Conceded

Implement `scored(T,W,S)` where S is the total number of goals scored by team T up to (and including) week W.
Implement `conceded(T,W,C)` where C is the total number of goals conceded by team T up to week W. Assume T and W are given as constants.

```
?- scored(juventus,5,S).
S = 9

?- conceded(juventus,5,C).
C = 8
```

Total Average of a Team

Implement `average(T,W,A)` where A is the average *(goals scored – goals conceded)* of a team T gathered up to (and including) week W. Assume T and W are given as constants.

```
?- average(kobenhavn, 3, A).
A = -6

?- average(kobenhavn, 6, A).
A = -9
```

0.4 Order and Top Three

Implement `order(L,W)` where W (week) is given as constant and league order in that week will be retrieved in L. Additionally, implement `topThree([T1,T2, T3],W)` where T1 T2 and T3 are the top teams when we are in the given week W. Assume that the order is decided according to **average** i.e *the one with the highest average will be at the top.* If the two teams have the same average then the order can be in any order.

```
?- order(L, 6).
L = [realmadrid, manutd, bdortmund, arsenal, fcnapoli, shaktard, juventus, bleverkusen,
galatasaray, realsociedad, kobenhavn, omarseille]

?-topThree(L, 6).
L = [realmadrid, manutd, bdortmund]
```

```
?- order([realmadrid, manutd, bdortmund, arsenal, fcnapoli, shaktard, juventus, bleverkusen,
galatasaray, realsociedad, kobenhavn, omarseille], W).
W = 6
```

Notes

- **IMPORTANT!!!** You may not use any of the following in any predicate you write:
  - `;` (or)
  - `->` (if-then)
  - `!` (cut)
  - `insert`
  - Any other operator that is not a part of "true" logic programing.
- You must put all your predicates in the same file. Do not create a different file for each predicate.
- Write comments in your code! It will be graded.
- Prepare a report file that explains our implementation.
- All the files will be submitted via Moodle.
- The project will be done individually, not as a group.
- We will test your program with some other database.
- Read the Programming Projects section in the document General Information for Students link in the Syllabus.