CMPE260.01
PRINCIPLES OF PROGRAMMING LANGUAGES
TUNGA GÜNGÖR

PROJECT-1
PROGRAMMING PROJECT

22.04.2018

STUDENT:
HASAN BASRİ BALABAN
2016400297

INTRODUCTION

In this project, we are asked to implement some prolog predicates to give information about some soccer teams and the matches that they have played.

We are given a knowledge base “cl_base.pl” which holds predicates about teams and the matches. The knowledge base has two kind of predicate, 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.”

We are asked for to implement several predicates by using built-in functions and logic programming. The predicates that we implement are:

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

“wins(T,W,L,N).” predicate 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).” predicate 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).” predicate implies that L involves the teams that team T could not defeat also did not lose to and N is the number of elements in L.

“scored(T,W,S).” predicate where S is the total number of goals scored by team T up to (and including) week W.

“conceded(T,W,C).” predicate where C is the total number of goals conceded by team T up to week W.

“average(T,W,A).” predicate where A is the average (goals scored - goals conceded) of a team T gathered up to (and including) week W.

“order(L,W).” predicate where W (week) is given as constant and league order in that week will be retrieved in L.

“topThree([T1,T2, T3],W).” predicate where T1 T2 and T3 are the top teams when we are in the given week W.

PROGRAM INTERFACE

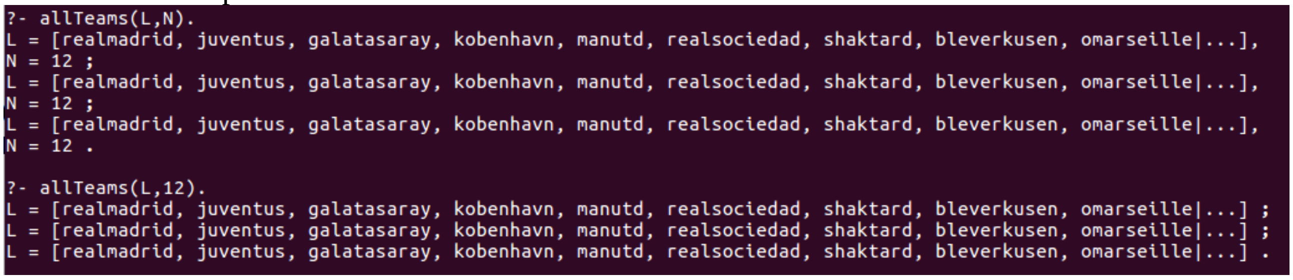
First thing that we need to run the program is installing swi-prolog from it’s official website. After the installation, we can run the program by following these steps :

1) Open a new terminal by pressing “Ctrl + Alt + T”
2) Then go to program’s directory using “cd” (change directory) command
3) Then type “prolog” to start the prolog interface and load the knowledge base of the program by typing “[*knowledge-base-file-name*].” or “consult(*knowledge-base-file-name).”*
4) After all of these steps, we can now use our implemented predicates;
   - allTeams(L,N)
   - wins(T,W,L,N)
   - losses(T,W,L,N)
   - draws(T,W,L,N)
   - scored(T,W,S)
   - conceded(T,W,C)
   - average(T,W,A)
   - order(L,W)
   - topThree([T1,T2,T3],W)
5) In the end, we can write “halt.” to exit prolog.

PROGRAM EXECUTION

In this section, the description of predicates and input-output relations topics will be covered.

allTeams predicate :

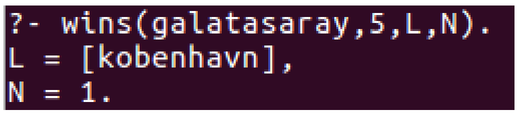


```
?- allTeams(L,N).
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille|...],
N = 12 ;
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille|...],
N = 12 ;
L = [realmadrid, juventus, galatasaray,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

?- allTeams(L,12).
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille|...] ;
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille|...] ;
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille|...] .
```

This predicate needs two input variables, first one represent a list which the list of all teams will be retrieved in, and the second parameter holds the number of elements inside the list. Predicate use permutation built-in function to give the list in it’s all kind of permutations.

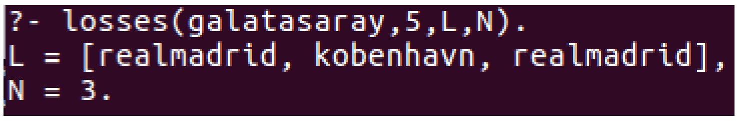
wins predicate:



```
?- wins(galatasaray,5,L,N).
L = [kobenhavn],
N = 1.
```

This predicate needs four parameters to execute, first parameter is the name of the team that we want to take information about, second parameter is a number represents a week. Third and fourth parameters are the outputs of the predicate. Third parameter holds a list of teams that defeated by the given team in or before the given week, and the fourth parameter is the number of teams in that list.

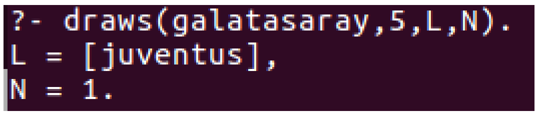
losses predicate:



```
?- losses(galatasaray,5,L,N).
L = [realmadrid, kobenhavn, realmadrid],
N = 3.
```

This predicate needs four parameters to execute, first parameter is the name of the team that we want to take information about, second parameter is a number represents a week. Third and fourth parameters are the outputs of the predicate. Third parameter holds a list of teams that beat the given team in or before the given week, and the fourth parameter is the number of teams in that list.

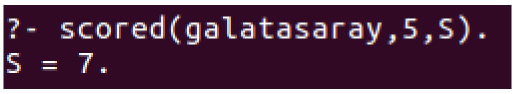
draws predicate:



```
?- draws(galatasaray,5,L,N).
L = [juventus],
N = 1.
```

This predicate needs four parameters to execute, first parameter is the name of the team that we want to take information about, second parameter is a number represents a week. Third and fourth parameters are the outputs of the predicate. Third parameter holds a list of teams that neither defeated by the given team nor beat the given team in or before the given week, and the fourth parameter is the number of teams in that list.

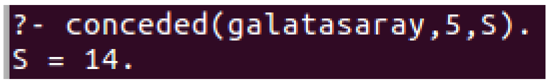
scored predicate:



```
?- scored(galatasaray,5,S).
S = 7.
```

This predicate needs three parameters to execute, first parameter is the name of the team that we want to take information about, second parameter is a number represents a week. Third parameter holds a number S where S is the total number of goals scored by the given team up to (and including) the given week W.

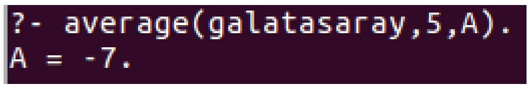
conceded predicate:



```
?- conceded(galatasaray,5,S).
S = 14.
```

This predicate needs three parameters to execute, first parameter is the name of the team that we want to take information about, second parameter is a number represents a week. Third parameter holds a number C where C is the total number of goals conceded by the given team up to (and including) the given week W.

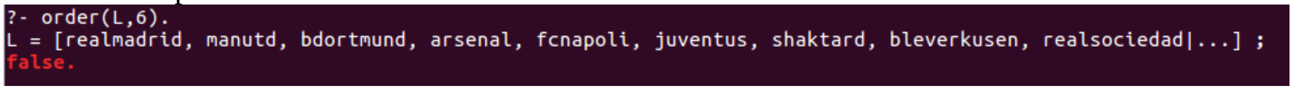
average predicate:



```
?- average(galatasaray,5,A).
A = -7.
```

This predicate needs three parameters to execute, first parameter is the name of the team that we want to take information about, second parameter is a number represents a week. Third parameter holds a number A where A is the average (goals scored - goals conceded) of the given team gathered up to (and including) the given week W.

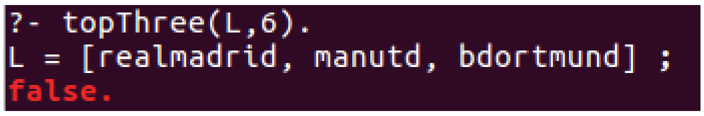
order predicate:



```
?- order(L,6).
L = [realmadrid, manutd, bdortmund, arsenal, fcnapoli, juventus, shaktard, bleverkusen, realsociedad|...] ;
false.
```

This predicate needs two parameters to execute, first parameter is a variable that will hold a list of the given week’s league order, and the second parameter is a number that represents the week that we want to take information about.

topThree predicate:



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

This predicate needs two parameters to execute, first parameter is a variable that will hold a list of the given week’s top three teams, and the second parameter is a number that represents the week that we want to take information about.

INPUT AND OUTPUT

The program reads a knowledge base from a .pl file which contains predicates about a league’s teams and matches. To construct such a knowledge base, we should write two kinds of predicates. First one is team(A,B). predicate where A is the team’s name and B is the team’s city. The second predicate is match(A,B,C,D,E). predicate where A is the week, B is the first team’s name, C is the first team’s score, D is the second team’s name and E is the second team’s score.

To load this knowledge base type [*knowledge-base*],[*predicates-file*]. inside the prolog. Then use the predicates that are explained in detail in Program Execution section.

PROGRAM STRUCTURE

```
/**
* allTeams predicate finds all the teams that given in the team predicates and returns a list that
contains all the teams.
* it uses findall function to search all the teams,
* then it uses permutation fuction to generate several versions of the team list,
* finally it uses the length function to return the number of teams in the list.
*/
allTeams(L,N):-
        findall(X,team(X,_),LTemp),
        permutation(LTemp,L),
        length(L,N).
/**
* 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.
* it uses findall function two times, in the first search it finds the given team's home matches, in the
second search it finds the given team's away matches,
* then it appends two temporary list to the final list (L), and uses length function to return the
number of teams in the list.
*/
wins(T,W,L,N):-
        findall(X, (match(A,T,B,X,D), A=<W, B>D), LTemp1),
        findall(Z, (match(C,Z,E,T,F), C=<W, F>E), LTemp2),
        append(LTemp1,LTemp2,L),
        length(L,N).
/**
* 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.
* it uses findall function two times, in the first search it finds the given team's home matches, in the
second search it finds the given team's away matches,
* then it appends two temporary list to the final list (L), and uses length function to return the
number of teams in the list.
*/
losses(T,W,L,N):-
        findall(X, (match(A,T,B,X,D), A=<W, B<D), LTemp1),
        findall(Z, (match(C,Z,E,T,F), C=<W, F<E), LTemp2),
        append(LTemp1,LTemp2,L),
        length(L,N).
/**
* 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.
* it uses findall function two times, in the first search it finds the given team's home matches, in the
second search it finds the given team's away matches,
* then it appends two temporary list to the final list (L), and uses length function to return the
number of teams in the list.
*/
draws(T,W,L,N):-
        findall(X, (match(A,T,B,X,D), A=<W, B=D), LTemp1),
        findall(Z, (match(C,Z,E,T,F), C=<W, F=E), LTemp2),
        append(LTemp1,LTemp2,L),
         length(L,N).
```

```
/**
* scored(T,W,S) implies that S is the total number of goals scored by team T up to (and including) week W.
* it uses findall function two times, in the first search it finds the given team's home matches, in the second search it finds the given team's away matches,
* then it appends two temporary list to the final list (L), and uses sumList predicate to sum up the elements of the list.
*/
scored(T,W,S):-
	findall(X, (match(A,T,X,_,_), A=<W), LTemp1),
	findall(Z, (match(C,_,_,T,Z), C=<W), LTemp2),
	append(LTemp1,LTemp2,L),
	sumList(L,S).
/**
* conceded(T,W,S) implies that S is the total number of goals conceded by team T up to (and including) week W.
* it uses findall function two times, in the first search it finds the given team's home matches, in the second search it finds the given team's away matches,
* then it appends two temporary list to the final list (L), and uses sumList predicate to sum up the elements of the list.
*/
conceded(T,W,C):-
	findall(X, (match(A,T,_,_,X), A=<W), LTemp1),
	findall(Z, (match(C,_,Z,T,_), C=<W), LTemp2),
	append(LTemp1,LTemp2,L),
	sumList(L,C).
/**
* average(T,W,A) implies that A is the average (goals scored - goals conceded) of a team T gathered up to (and including) week W.
* it uses findall function two times, in the first search it finds the given team's home matches, in the second search it finds the given team's awat matches,
* then it appends two temporary list to the final list (L), and uses sumList predicate to sum up the elements of the list.
*/
average(T,W,A):-
	findall(X, (match(A,T,Sc,_,Co), A=<W,  X is Sc-Co), LTemp1),
	findall(Z, (match(C,_,Con,T,Sco), C=<W, Z is Sco-Con), LTemp2),
	append(LTemp1,LTemp2,L),
	sumList(L,A).
/**
* order(L,W) implies that W (week) is given as constant and league order in that week will be retrieved in L.
* it uses allTeamsNoPermu predicate to find a list of all teams in the team predicates,
* then it uses mergesort predicate to sort the finded teams acording to their averages in the given week,
* finally it uses reverse function to reverse the list, because the mergesort predicate returns the list in the ascending order.
*/
order(L,W):-
	allTeamsNoPermu(LTemp),
	mergesort(LTemp,LTemp2,W),
	reverse(LTemp2,L).
```

```prolog
/**
* topThree([T1,T2, T3],W) implies that T1 T2 and T3 are the top teams when we are in the given
week W.
* it uses order predicate to find a list of teams which are ordered according to their averages in the
given week,
* then it takes the first three element from the list and declares the given variables T1, T2 and T3.
*/
topThree([T1,T2,T3],W):-
	order(L,W),
	L = [H1|Tail1], T1 = H1,
	Tail1 = [H2|Tail2], T2 = H2,
	Tail2 = [H3|_], T3 = H3.
/**
* allTeamsNoPermu is a helper predicate for order predicate,
* it finds all the teams in the team predicates and returns the list (L).
* it just uses a findall function to search the team predicates.
*/
allTeamsNoPermu(L):-
	findall(X,team(X,_),L).
/**
* sumList predicate is another helper predicate.
* basically, it sums up the elements of a given list.
*/
sumList([], 0).
sumList([H|T], Sum):-
	sumList(T, Rest),
	Sum is H + Rest.
/**
* mergesort predicate is a helper predicate, too.
* it sort a given list in ascending order.
* it splits the given list recursively, until all the lists have only one element inside,
* then it merges the splitted lists back, and it merges them according to their averages in the given
week.
*/
mergesort([],[],_).
mergesort([A],[A],_).
mergesort([A,B|R],S,W):-
	split([A,B|R],L1,L2),
	mergesort(L1,S1,W),
	mergesort(L2,S2,W),
	merge(S1,S2,S,W).
/**
* split predicate is a helper predicate for mergesort predicate,
* it splits a list in to two piece.
*/
split([],[],[]).
split([A],[A],[]).
split([A,B|R],[A|Ra],[B|Rb]):-
	split(R,Ra,Rb).
```

```
/**
*merge predicate is also a helper predicate for mergesort predicate,
*it merges the splitted pieces of the list.
*/
merge(A,[],A,_).
merge([],B,B,_).
merge([A|Ra],[B|Rb],[A|M],W):-
        average(A,W,A1),
        average(B,W,A2),
        A1 =< A2,
        merge(Ra,[B|Rb],M,W).
merge([A|Ra],[B|Rb],[B|M],W):-
        average(A,W,A1),
        average(B,W,A2),
        A1 > A2,
        merge([A|Ra],Rb,M,W).
```

EXAMPLES

```
?- 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

?- wins(galatasaray,4,L,N).
L = [kobenhavn]
N = 1 ;
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
?- scored(juventus,5,S).
S = 9

?- conceded(juventus,5,C).
C = 8
?- average(kobenhavn, 3, A).
A = -6

?- average(kobenhavn, 6, A).
A = -9
?- order(L, 6).
L = [realmadrid, manutd, bdortmund, arsenal, fcnapoli, shaktard, juventus, bleverkusen,
galatasaray, realsociedad, kobenhavn, omarseille]

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

DIFFICULTIES ENCOUNTERED

The only difficulty that I face with while writing this program, is finding the necessary built-in functions. After I found the necessary functions and algorithms, program runs very well in the first time.

CONCLUSION

This project was about writing predicates by using logic programming and prolog. We wrote nine predicates to perform such things like listing all teams or a teams average etc. The project was very didactic for us.