|  |
| --- |
|  |
| CMPE260 PRINCIPLES OF PROGRAMMING LANGUAGES |
| REPORT OF THE PROJECT |
| **This project is about the knowledge of logic programming and Prolog.The task is to write Prolog predicates (statements) for manipulating lists of football teams,**  **and the matches played between them.**    **PROLOG ASSIGNMENT** |

**2016400357 YUSUF KALAYCI**

**22.04.2018**

**SUBMITTED TO: TEACHING ASSISTANT HASAN FERİT ENİŞER**

**CMPE 260 PRINCIPLES OF PROGRAMMING LANGUAGES**

**Prolog predicates (statements) for manipulating lists of football teams,and the matches played between them.**

**Introduction**

This project our main goal is using the Prolog Programming Languages and test the knowledge of logic programming and Prolog. The project includes predicates.pl file and inside it the Prolog predicates (statements) for manipulating lists of football teams,and the matches played between them can find. The cl\_base.pl is the database of our project.

**Program Interface**

This project made with the help of SWI-Prolog and Prolog language.First of all the user should download the SWI-Prolog in the official page.If the user like me use Linux-Ubuntu operating system then user should add including package in the terminal screen.Mac or windows users can watch tutorials about prolog installation. Program can learn with the help of watching videos or reading documents.

**Program Execution**

User manual part for user;

Please before the execution part follow the Program Interface Part and installation of prolog.

1.Open the terminal.(Ctrl+Alt+T)

2.Select working directory for example ;

byklyci $-cd Documents/Prolog Examples/

3.After the reach working directory you should write :

byklyci :~/Documents/Prolog Examples $-prolog

Then SWI-Prolog will start for execution.

4. You should write knowledge base for example;

?- [predicates],[cl\_base].

true

5.The knowledge base is opened and return true then the predicates can be try with the order.The main idea is ask prolog and it will answer the question with the help of predicates.For example;

?- average(kobenhavn,6,A).

A=-9.

The execution is user friendly for all user.

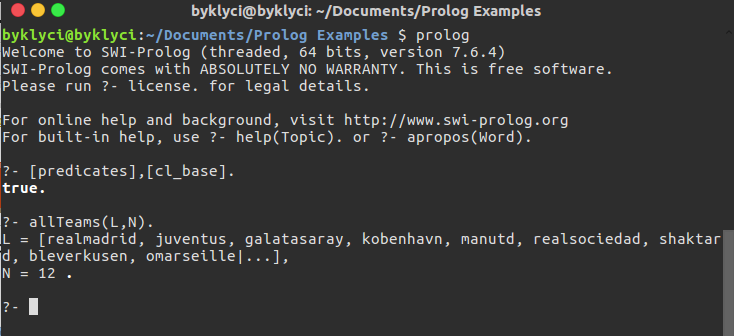
**Input and Output**

Our input is cl\_base.pl because this file contains database of the teams and matches between them. And also our predicates.pl file(it includes inside it the statements and our task about the project) The prolog is user friendly as I said before, so we can concatenate the . pl file at the same time ;

?- [predicates],[cl\_base].

true

Then we can say our output is the terminal.We can think like that ask prolog and see the answer.



**Program Structure**

First of all ı made some research about prolog languages.Then ı find some builtin functions for reduce my difficulties such as findall/2,findall/3,append/3.And also use the prolog language variables,sort algoritms.The main thing is the logic behind the prolog because when I understood exactly the logic, the program became easier. The predicates are;

?-allTeams(L,N).

?-wins(T,W,L,N).

?-losses(T,W,L,N).

?-draws(T,W,L,N).

?-scored(T,W,S).

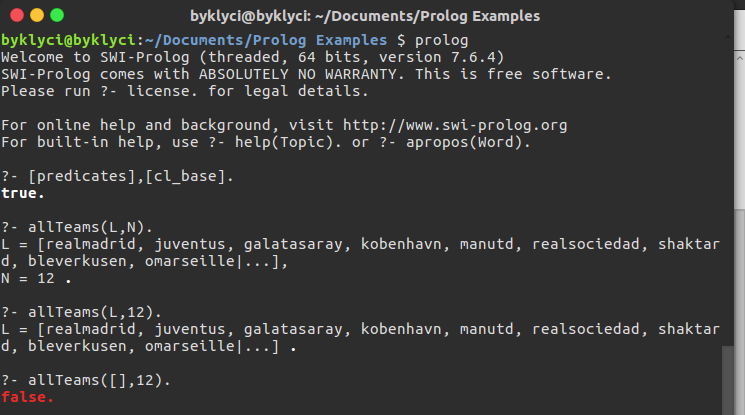
?-conceded(T,W,C).

?-average(T,W,A).

?-order(L,W).

?-topThree(L,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]

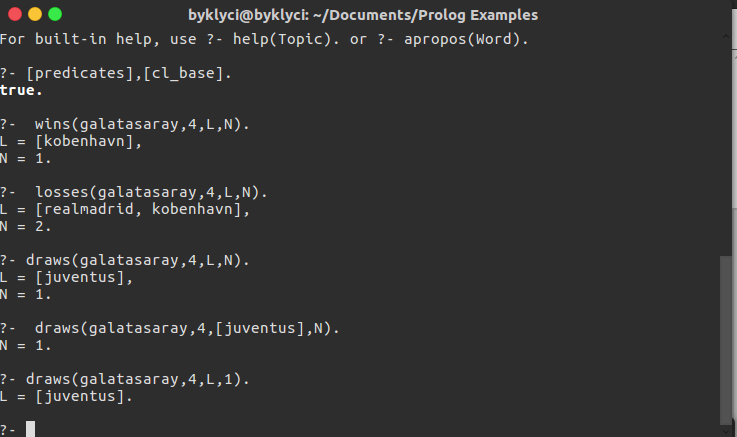
True

?- 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

?- 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

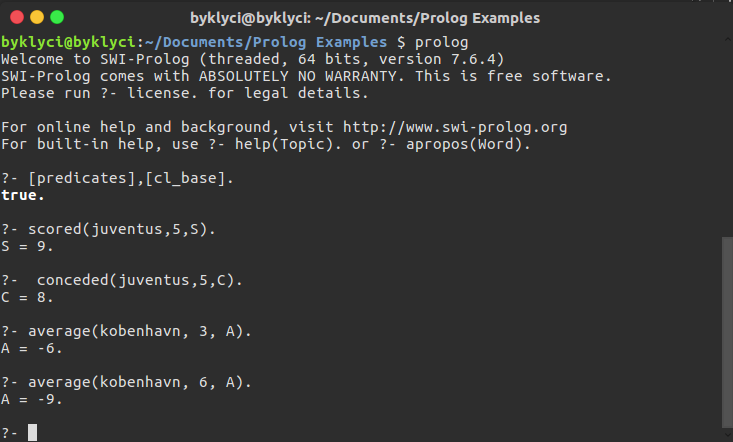
?- draws(galatasaray,4,L,1).

L = [juventus] ;

False

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

True



?- scored(juventus,5,S).

S = 9

?- conceded(juventus,5,C).

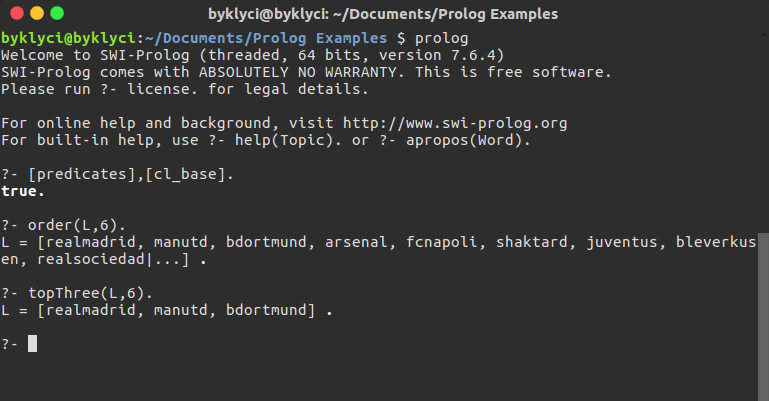
C = 8

?- 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]

?- order([realmadrid, manutd, bdortmund, arsenal, fcnapoli, shaktard, juventus, bleverkusen,

galatasaray, realsociedad, kobenhavn, omarseille], W).

W = 6

**Improvements and Extensions**

My program is so understandable,I used comment lines for the predicates and what I did in my program. The people who want to change or improve my program can be find the code in GitHub.

My Github account is:

byklyci

My codes and the my logic can improve or extend with the other usage of the other projects but it is enough for this project in my opinion.

**Difficulties Encountered**

First of all, the prolog language is the new language for me so I encounter some problem with the logic.Then I watch some videos, follow some tutorials and then I started the project.When I started , I also meet some problems but I search my faults on the internet. The order() predicate part is the most challenging part for me but I solved the problems then I finished my project.

**Conclusion**

Short for Programming Logic, Prolog is a [high-level programming language](https://www.webopedia.com/TERM/H/high_level_language.html) based on formal logic. Unlike traditional programming languages that are based on performing sequences of [commands](https://www.webopedia.com/TERM/C/command.html), Prolog is based on defining and then solving logical formulas. Prolog is sometimes called a declarative language or a rule-based language because its [programs](https://www.webopedia.com/TERM/P/program.html) consist of a list of facts and rules. Prolog is used widely for [artificial intelligence](https://www.webopedia.com/TERM/A/artificial_intelligence.html) applications, particularly [expert systems](https://www.webopedia.com/TERM/E/expert_system.html).

This is very helpful language and efficient homework for me.Thanks to Professor Tunga Güngör and also assistants Hasan Ferit Enişer and Özlem Şimşek for intoducing me with such an excellent programming language.

**Appendices**

http://SWI-Prolog.com/

https://www.webopedia.com/TERM/P/Prolog.html

Here is the my code :

/\*\*

\*April 2018

\* Implemented by Yusuf Kalaycı for the Bogazici University CMPE260 course project. All the codes are implemented by me,there are no copy-paste or stolen codes here, all the implementation is based on the knowledge that I obtained in the PS section and on the internet.Thanks to Professor Tunga Güngör and also assistants Hasan Ferit Enişer and Özlem Şimşek for intoducing me with such an excellent programming language.

\*

\*

\*/

/\*\*

\* allTeams() is the first predicate in this part findall3 used and the length of the list is

\* written on the terminal screen with the help of permutation of the all teams in the cl\_base.pl

\*/

allTeams(L,N):-

findall(X,team(X,\_),L1),

length(L1,N),permutation(L1,L).

/\*\*

\* wins() is the predicate that includes four variables and the main logic is finding weeks until the

\* wanted week.The team can play other teams in the home or away so there are two findall for two list

\* the append() is used for concenate two list together.I take the integer of score that more than

\* other team.So the team wins.

\*/

wins(T,W,L,N):-

findall(D, (match(Weeks\_of,T,C,D,E),Weeks\_of=<W,C>E),L1),

findall(B, (match(Weeks\_of,B,C,T,E),Weeks\_of=<W,E>C),L2),

append(L1,L2,L),

length(L,N).

/\*\*

\* losses() is the predicate that includes four variables, the main logic is finding weeks until the

\* wanted week.The team can play other teams in the home or away so there are two findall for two list

\* the append() is used for concenate two list together.I take the integer of score that less than

\* other team.So the team losses.

\*/

losses(T,W,L,N):-

findall(D,(match(A,T,C,D,E),A=<W,C<E),L1),

findall(B,(match(A,B,C,T,E),A=<W,C>E),L2),

append(L1,L2,L),

length(L,N).

/\*\*

\* draws() is the predicate that include four variable and the main logic is finding weeks until the

\* wanted week.The team can play other teams in the home or away so there are two findall for two list

\* the append() is used for concenate two list together.I take the integer of score that equal the

\* other team score.So the teams draw with each other.

\*/

draws(T,W,L,N):-

findall(D,(match(A,T,C,D,E),A=<W,C=E),L1),

findall(B,(match(A,B,C,T,E),A=<W,C=E),L2),

append(L1,L2,L),

length(L,N).

/\*\*

\* sum\_list(), sum the list and write the results.

\*/

sum\_list([],0).

sum\_list([C|E],S):-

sum\_list(E,Rest),

S is C + Rest.

/\*\*

\* scored() is the predicate that includes three variables and it takes the selected team score

\* which means that the goal against other team inside wanted weeks.

\* The team can play other teams in the home or away so there are two findall for two list

\* the append() is used for concenate two list together.sum\_list() predicate calls here to sum the

\* each week score for specified team.

\*/

scored(T,W,S):-

findall(C,(match(A,T,C,\_,\_),A=<W),L1),

findall(E,(match(A,\_,\_,T,E),A=<W),L2),

append(L1,L2,L),

sum\_list(L,S).

/\*\*

\* conceded() is the predicate that includes three variables and it takes the selected team conceded

\* which means that other teams goal against selected team inside wanted weeks.

\* The team can play other teams in the home or away so there are two findall for two list

\* the append() is used for concenate two list together.sum\_list() predicate calls here to sum the

\* each week concede a goal for specified team.

\*/

conceded(T,W,C):-

findall(C,(match(A,\_,C,T,\_),A=<W),L1),

findall(E,(match(A,T,\_,\_,E),A=<W),L2),

append(L1,L2,L),

sum\_list(L,C).

/\*\*

\* average() is the predicate that includes three variables and it takes the selected team score

\* and also concede a goal inside wanted weeks.Average means that goal score-goal conceded.

\* The team can play other teams in the home or away and there are four findall for score predicate and

\* the concede goal predicate.The append() is used for concenate two list together.

\* sum\_list() predicate calls here to sum the lists then after the sum; the average can find

\* with A is Src - Cnc.

\*/

average(T,W,A):-

findall(Cs,(match(Xw,T,Cs,\_,\_),Xw=<W),L1),

findall(Es,(match(Xq,\_,\_,T,Es),Xq=<W),L2),

append(L1,L2,L3),

sum\_list(L3,Scr),

findall(Cc,(match(Xe,\_,Cc,T,\_),Xe=<W),L4),

findall(Ec,(match(Xr,T,\_,\_,Ec),Xr=<W),L5),

append(L4,L5,L6),

sum\_list(L6,Cnc),

A is Scr - Cnc.

/\*\*

\* insertion\_sort () is the predicate that helps the order() predicate.It sort

\* a given list in an orderly and use accumulator when doing it.It make insertion

\* into the the list and inside the insertion\_sort() the average() predicate calls

\* because the sorting should be done with the help of it.

\*/

insertion\_sort(List,Sorted,W):-i\_sort(List,[],Sorted,W).

i\_sort([],Acc,Acc,\_).

i\_sort([H|T],Acc,Sorted,W):-insertion(H,T,NAcc,M,W),i\_sort(NAcc,[M|Acc],Sorted,W).

insertion(X,[],[],X,\_).

insertion(X,[Y|T],[Y|NAcc],M,W):-average(X,W,A1), average(Y,W,A2), (A1>A2),insertion(X,T,NAcc,M,W).

insertion(X,[Y|T],[X|NAcc],M,W):-average(X,W,A1), average(Y,W,A2), (A1=<A2),insertion(Y,T,NAcc,M,W).

insertion(X,[],[X],\_).

/\*\*

\* order() is the predicate that implemented order(L,W) where W (week) is given as

\* constant and league order in that week will be retrieved in L.

\* 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.In order() predicate insertion\_sort(),reverse() and findall()

\* of all team is used.

\*/

order(L,W):-

findall(X,team(X,\_),L1),

insertion\_sort(L1,LA,W),

reverse(LA,L).

/\*\*

\* topThere() is the predicate that implement topThree([T1,T2, T3],W) where T1T2 and

\* T3 are the top teams when we are in the given week W.When implementing this

\* predicate call order() predicates insiders.Then delete the athor elements in the L

\* except first three of them.

\*/

topThree(L,W):-

findall(X,team(X,\_),L1),

insertion\_sort(L1,LA,W),

delete([realmadrid,manutd,bdortmund,\_,\_,\_,\_],LA,L).