U18CO018

Shekhaliya Shubham

PPL
Assignment 1

1) Write a program in Prolog that uses following predicates

Write, nl, read, consult,halt,statistics.

Code:

```
count :-
  write('Write a number: '),
  read(Number),
  process(Number).
process(time) :- statistics, nl, count.
process(exit) :- halt.
process(stop) :- !.
process(Number) :-
  C is Number * Number,
  write('Square of '),write(Number),write(': '),write(C),nl, count.
```

```
?- consult('E:/Asem7/PPL/Assignment1/first.pl').
true.

?- count.
Write a number: 10.
Square of 10: 100
Write a number: |: 4.
Square of 4: 16
Write a number: |: time.

% Started at Thu Aug 05 23:15:05 2021
% 2.234 seconds cpu time for 539,323 inferences
% 10,809 atoms, 6,918 functors, 5,290 predicates, 130 modules, 343,842 VM-codes
%
%                 Limit   Allocated      In use
% Local  stack:          -      20 Kb    2,280  b
% Global stack:          -     124 Kb       77 Kb
% Trail  stack:          -      30 Kb    1,192  b
%       Total:    1,024 Mb     174 Kb       80 Kb
%
% 3 garbage collections gained 418,024 bytes in 0.000 seconds.
% 29 atom garbage collections gained 6,707 atoms in 0.000 seconds.
% 34 clause garbage collections gained 1,729 clauses in 0.000 seconds.
% Stack shifts: 4 local, 5 global, 5 trail in 0.000 seconds
% 3 threads, 0 finished threads used 0.000 seconds

Write a number: |:
```

2) Try to answer the following questions first “by hand” and then verify your answers using a Prolog interpreter.

(a) Which of the following are valid Prolog atoms?

f, loves(john,mary), Mary, _c1, 'Hello', this_is_it

Ans:

Atoms are usually strings made up of lower- and uppercase letters, digits, and the underscore, starting with a lowercase letter.

f, this_is_it are atoms

(b) Which of the following are valid names for Prolog variables?

a, A, Paul, 'Hello', a_123, _, _abc, x2

Ans: Variables are strings of letters, digits, and the underscore, starting with a capital letter or an underscore.

A, Paul, _, _abc are variable

(c) What would a Prolog interpreter reply given the following query?

?- f(a, b) = f(X, Y).

```
?- f(a, b) = f(X, Y).
X = a,
Y = b.
```

(d) Would the following query succeed?

?- loves(mary, john) = loves(John, Mary).

Why?

```
?- loves(mary, john) = loves(John, Mary).
John = mary,
Mary = john.
```

Because of query matching rules.

(e) Assume a program consisting only of the fact

a(B, B).

has been consulted by Prolog. How will the system react to the following query?

?- a(1, X), a(X, Y), a(Y, Z), a(Z, 100).

Why?

Predicate a(B,B) means both the arguments are same

So

a(1, X) -> X = 1

a(X, Y) -> X = Y

a(Y, Z) -> Y = Z

a(Z, 100) -> Z = 100

```
?- consult('E:/Asem7/PPL/Assignment1/try.pl').
true.

?- a(X, Y).
X = Y.

?- a(Y, Z).
Y = Z.

?- a(1, X).
X = 1.

?- a(Z, 100).
Z = 100.
```

3) Read the section on matching again and try to understand what's happening when you submit the following queries to Prolog.

(a) ?- myFunctor(1, 2) = X, X = myFunctor(Y, Y).

```
?- myFunctor(1, 2) = X, X = myFunctor(Y, Y).
false.
```

Because when match both the time values of X are different because the value of Variable Y is contradic

(b) ?- f(a, _, c, d) = f(a, X, Y, _).

```
?- f(a, _, c, d) = f(a, X, Y, _).
Y = c.
```

Because number of arguments and predicates are same so try to match each argument

(c) ?- write('One '), X = write('Two ').

```
?- write('One '), X = write('Two ').
One
X = write('Two ').
```

Because first stamen write while second is matching statement and X is complex variable so it assign value to that

4) Draw the family tree corresponding to the following Prolog program:

```
female(mary).
female(sandra).
female(juliet).
female(lisa).
male(peter).
male(paul).
male(dick).
male(bob).
male(harry).
parent(bob, lisa).
parent(bob, paul).
parent(bob, mary).
parent(juliet, lisa).
parent(juliet, paul).
parent(juliet, mary).
parent(peter, harry).
```

parent(lisa, harry).

parent(mary, dick).

parent(mary, sandra).

After having copied the given program, define new predicates (in terms of rules using male/1, female/1 and parent/2) for the following family relations:

(a) father

(b) sister

(c) grandmother

(d) cousin

You may want to use the operator \=, which is the opposite of =. A goal like X \= Y succeeds, if the two terms X and Y cannot be matched.

Example: X is the brother of Y, if they have a parent Z in common and if X is male and if X and Y don't represent the same person. In Prolog this can be expressed through the following rule:

```
brother(X, Y) :-
parent(Z, X),
parent(Z, Y),
male(X),
X \= Y.
```

Code:

```
female(mary).
female(sandra).
female(juliet).
female(lisa).
male(peter).
male(paul).
```

```
male(dick).
male(bob).
male(harry).
parent(bob, lisa).
parent(bob, paul).
parent(bob, mary).
parent(juliet, lisa).
parent(juliet, paul).
parent(juliet, mary).
parent(peter, harry).
parent(lisa, harry).
parent(mary, dick).
parent(mary, sandra).

father(X, Y) :-
        male(X),
        parent(X, Y).

sister(X, Y) :-
        female(X),
        parent(Z, X),
        parent(Z, Y),
        X \= Y.

grandmother(X, Y) :-
        female(X),
        parent(X, Z),
        parent(Z, Y).

siblings(X, Y) :-
        parent(Z, X),
```

```
        parent(Z, Y),
        X \= Y.

cousin(X, Y) :-
        parent(Z, X),
        parent(W, Y),
        siblings(Z, W),
        X \= Y.
```

Snap

```
?- consult('E:/Asem7/PPL/Assignment1/family.pl').
true.

?- father(bob, paul).
true .

?- father(bob, X).
X = lisa ;
X = paul ;
X = mary.

?-
```

```
?- sister(X, Y).
X = mary,
Y = lisa .

?- sister(mary, Y).
Y = lisa .
```

```
?- grandmother(X, Y).
X = juliet,
Y = harry ;
X = juliet,
Y = dick .

?- grandmother(X, harry).
X = juliet .



?- cousin(harry, Y).
Y = dick .

?- cousin(X, sandra).
X = harry .

?-
```