PPL LAB 1

SAHIL BONDRE: U18CO021

1) Write a program in Prolog that uses following predicates Write, nl, read, consult,halt,statistics.

```
say_hi() :-
  write('Please enter your name: '),
  nl,
  read(Name),
  atom_concat('Hello, ', Name, Greeting),
  write(Greeting).


animal(elephant).
animal(dog).
animal(cat).


count_animals(Count) :- aggregate_all(count, animal(_), Count).
```

```
?- consult('q1.pl').
true.

?- say_hi.
Please enter your name:
|: sahil.
Hello, sahil
true.

?- statistics.
% Started at Fri Jul 30 23:10:06 2021
% 0.088 seconds cpu time for 259,240 inferences
% 5,622 atoms, 4,130 functors, 2,928 predicates, 40 modules, 105,957 VM-codes
%
%                     Limit   Allocated      In use
% Local  stack:           -       20 Kb    2,128  b
% Global stack:           -       64 Kb       28 Kb
% Trail  stack:           -       34 Kb    1,600  b
%        Total:    1,024 Mb      118 Kb       31 Kb
%
% 2 garbage collections gained 166,736 bytes in 0.000 seconds.
% 3 atom garbage collections gained 718 atoms in 0.003 seconds.
% 5 clause garbage collections gained 118 clauses in 0.000 seconds.
% Stack shifts: 2 local, 3 global, 3 trail in 0.000 seconds
% 2 threads, 0 finished threads used 0.000 seconds
true.

?- count_animals(X).
X = 3.

?- halt.
➜  lab-01 git:(master) x
```

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

```
?- atom(f).
true.

?- atom(loves(john, mary)).
false.

?- atom(Mary).
false.

?- atom(_c1).
false.

?- atom('Hello').
true.

?- atom(this_is_it).
true.

?-
```

(b) Which of the following are valid names for Prolog variables?
a, A, Paul, 'Hello', a_123, _, _abc, x2

```
?- a = 'var'.
false.

?- A = 'var'.
A = var.

?- 'Hello' = 'var'.
false.

?- a_123 = 'var'.
false.

?- _ = 'var'.
true.

?- _abc = 'var'.
_abc = var.

?- x2 = 'var'.
false.

?-
```

(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?

Yes, it will succeed as the same function is used and thus the following output will be generated:

John = mary

Mary = john

LHS parameters are atoms, while RHS are variables

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

?-
```

(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?

The fact equates the two variables so will be the case for the rest of the variables.

```
?- consult('q2.pl')
|    .
true.

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

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

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

?- 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).
   (b) ?- f(a, _, c, d) = f(a, X, Y, _).
   (c) ?- write('One '), X = write('Two ').

Here, the function used is the same hence and is connected to the same variable X. However, myFunctor(Y, Y) suggests that both params be same but in myFunctor(1, 2), it is different thus results in false:

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

Here, the function used is the same thus being equalized. When we equate both functions, we can come to the conclusion that Y = c:

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

write('One ') writes within the console and is the true statement. It being followed by a comma suggests that AND operator which is followed by another correct statement hence written within the console too:

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

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).

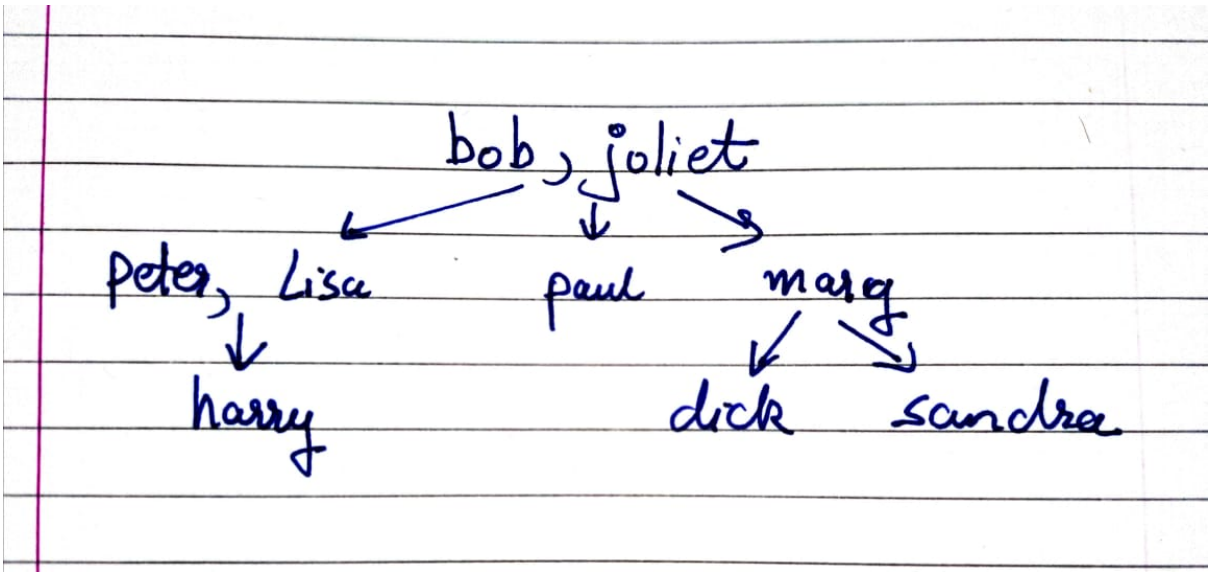
parent(juliet, mary).

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.

```
% father relation
father(X,Y) :- parent(X,Y), male(X).


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


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


% grandparent relation
grandparent(X, Y) :- parent(X, Z), parent(Z, Y).


% cousin relation
```

```
cousin(X,Y) :- parent(X, Z), parent(Y, S), Z \= S, parent(Z, A), parent(S, A),
X \= Y.
```

```
?- father(X, paul).
X = bob .

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

?- consult('q4.pl').
true.

?- father(X, paul).
X = bob .

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

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

?- cousin(X, Y).
false.
```