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

CODE:

```
run():-write("Hello, Name: "), read(N),nl,

write("Hey there "),write(N),nl,
write("Statistics: "),nl,
statistics.
```

```
?- run.
naman
|: .
Hello, Name:
Hey there naman
Statistics:
% Started at Mon Mar 21 11:52:03 2022
% 0.189 seconds cpu time for 428,475 inferences
% 5,872 atoms, 3,620 functors, 2,725 predicates, 51 modules, 99,173 VM-codes
%
%                         Limit    Allocated       In use
% Local  stack:    268,435,456      126,976        1,912 Bytes
% Global stack:    268,435,456      126,960        5,824 Bytes
% Trail  stack:    268,435,456      129,016          648 Bytes
%
% 4 garbage collections gained 155,496 bytes in 0.001 seconds.
% 2 clause garbage collections gained 73 clauses in 0.000 seconds.
% Stack shifts: 4 local, 5 global, 4 trail in 0.001 seconds
% 2 threads, 0 finished threads used 0.000 seconds
true.
```

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

Atom:- a, ’Hello’, a_123, x2
Variables:- A, Paul, _, _abc,

```
?- var(a).
false.

?- var(A).
true.

?- var(Paul).
true.

?- var('Hello').
false.

?- var(a).
false.

?- var(a_123).
false.

?- var(_).
true.

?- var(_abc).
true.

?- var(x2).
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?

Since, John, Mary are variables and does not hold any value. So, mary and john which are atoms, are assigned to John and Mary respectively.

```
?- 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?

a(B,B) fact returns true when both arguments are equal. So, after a(1,X), X=1. Then, a(X,Y) assigns Y=1. Now a(Y,Z) assigns Z=1. In last a(Z,100), Since Z has already value which is 1, So it will compare 1 and 100. Since they are not equal, it will result false.

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

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).

Here, X will have value myFunctor(1,2). But when second time X=myFunctor(Y,Y) is called it checks as already x holds value. Now 1 is not equal to 2 hence myFunctor is not equal to myFunctor(Y,Y). So false.

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

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

Here, f(a, _, c, d) = f(a, X, Y, _) will check first both first variable. Since they are equal it will go for second variable. Here, _ is used which is a universal variable symbol. Hence it will return true and X also doesn't hold any value it will not show in output. In third variable, one is atom and other is variable which holds no value. Hence it will assign Y=c which will be printed. In fourth variable '_' is used, which is true. Hence, output will be printed as Y=c.

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

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

So, first write will print One. X= write('Two ') will assign fact to variable X. Hence it will print X and its value.

```
?- 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.
```

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).

% Relationships
father_of(X,Y):-male(X),parent(X,Y).
siblings(X,Y):-parent(Z,X),parent(Z,Y),X\==Y.
sister(X,Y):-siblings(X,Y),female(Y),X\==Y.
grandmother(X,Y):-parent(X,Z),parent(Z,Y),female(X).
cousin(A,B):- parent(X,A),parent(Y,B),siblings(X,Y)
```

```
?- father_of(X, mary).
X = bob .

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

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

?- cousin(juliet, X).
false.

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

?-
```