Principles of Programming Language (CS302)

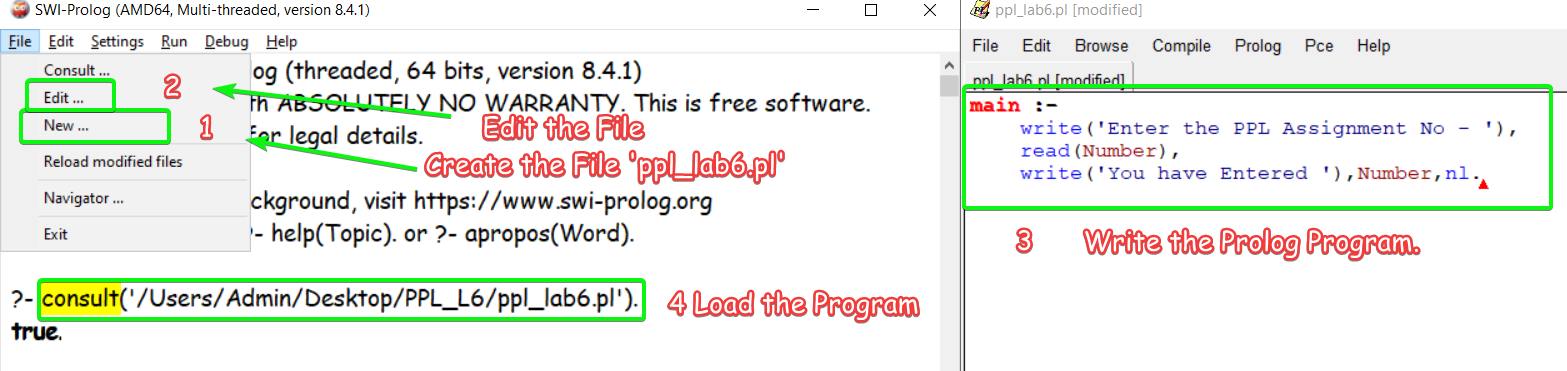
Assignment - 6

**U19CS012**

1.) Write a Program in Prolog that uses following Built-in Predicates

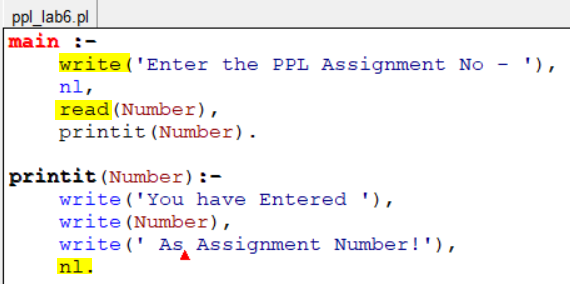
(a) **File Loading**

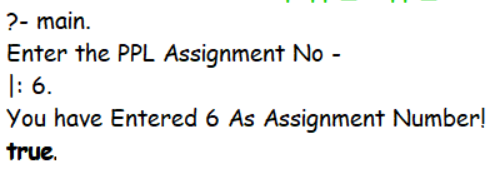
**consult(F)** -> Loads program from the file F.



(b) **Input predicates**

**read(X)** -> Read one clause from the current input and unify it with X. If there is no further input, X is unified with end\_of\_file.

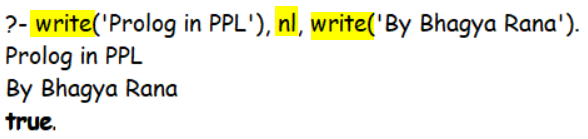




(c) **Output predicates**

**write(X)** -> Write the single value X to the current output file.

**nl** -> Write a newline to the current output file.

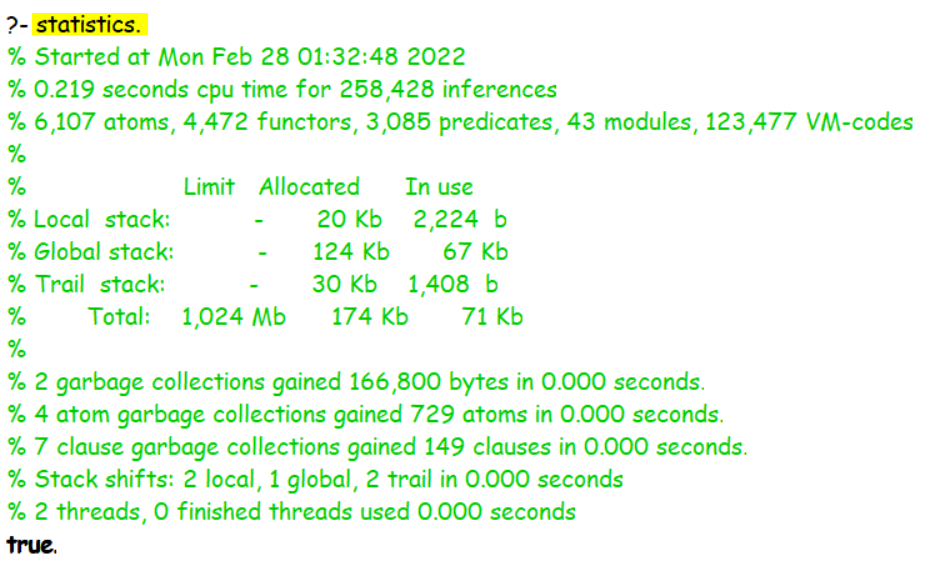


(d) **Utility Functions**

**halt** -> causes the Prolog system to terminate



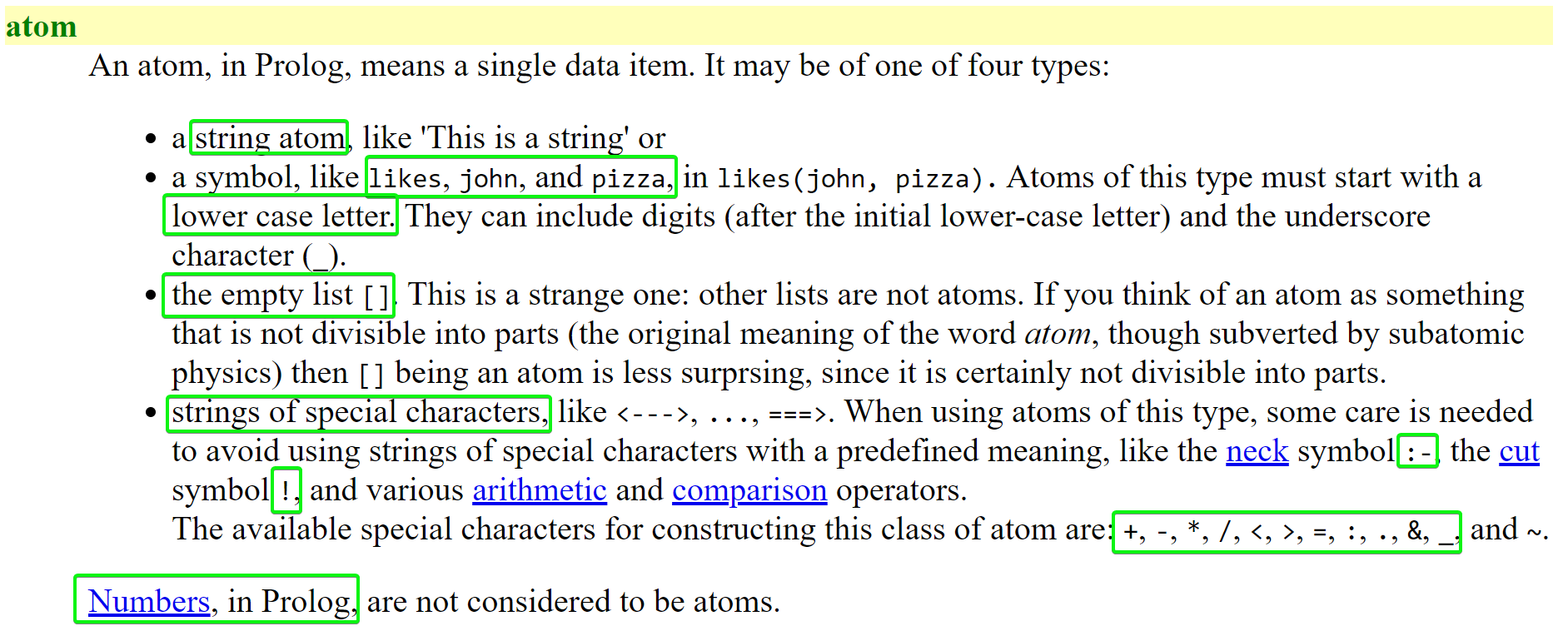
**statistics** -> prints system statistics.

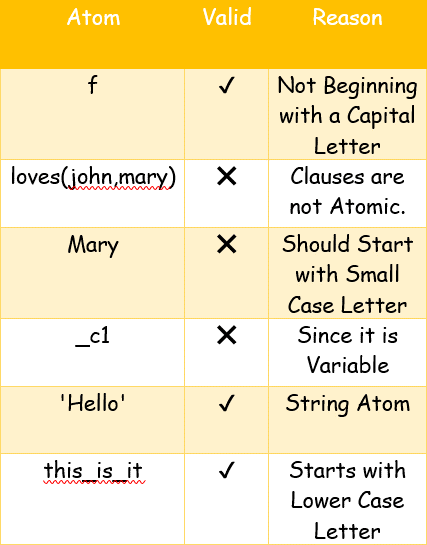
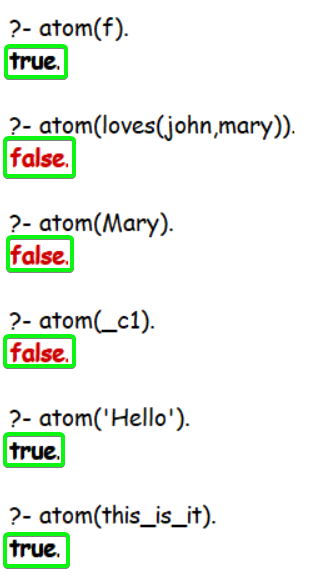


2.) Try to answer the following questions first “by hand” and then verify your answers using a Prolog interpreter.

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

An atom, in Prolog, means a **Single Data item**.



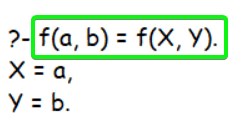
 

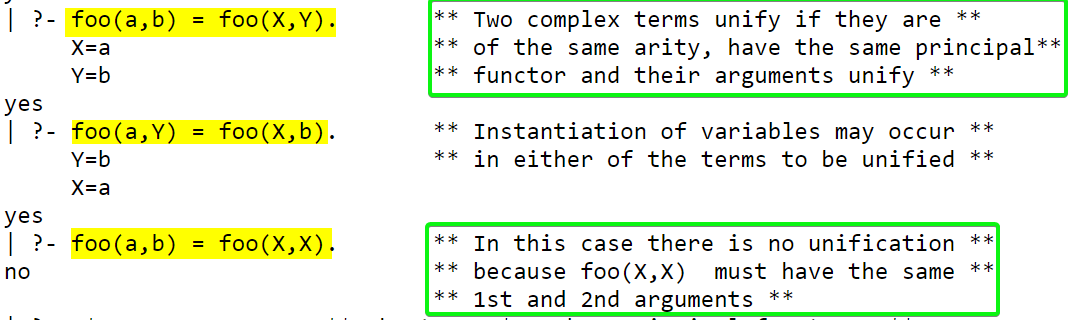
(b) Which of the following are valid names for Prolog variables?

**A variable** is written as a sequence of letters and digits, beginning with a capital letter. The **underscore (\_)** is considered to be a **capital letter.**

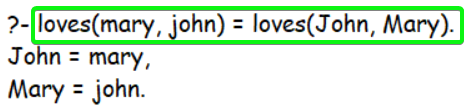
|  |  |  |
| --- | --- | --- |
| Variable | Valid | Reason |
| a | ❌ | Not Beginning with a Capital Letter |
| A | ✔️ |  |
| Paul | ✔️ |  |
| ‘Hello’ | ❌ | Not Beginning with a Capital Letter |
| a\_123 | ❌ | Not Beginning with a Capital Letter |
| \_abc | ✔️ |  |
| x2 | ❌ | Not Beginning with a Capital Letter |

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





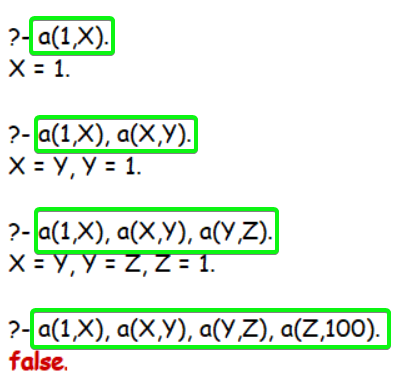
(d) Would the following query succeed? ?- loves(mary, john) = loves(John, Mary). – YES



Why? Since **J**ohn is Variable that is Unified with Value mary and **M**aryis unifies with Value 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). - false



Why?

– Since 1 = X = Y = Z & Z = 100 are contradictory.

It is important to note that the same variable has to be instantiated with the same

value **throughout an expression.**

The only exception to this rule is the anonymous variable **\_**, which is considered to be unique whenever it occurs.

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

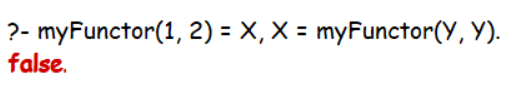
* Two terms are said to **match** if they are either identical or if they can be made identical by means of variable instantiation.
* Instantiating a variable **means** assigning it a fixed value.
* Two free variables also match, because they could be instantiated with the same ground term.

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

The Function ‘myFunctor’ used is same and matched to same variable ‘X’ [Fixed].

However, myFunctor(Y, Y) suggest that both the parameters be same, but that is not the case with myFunctor(1,2).

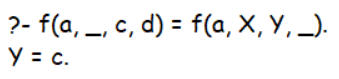
Therefore the Goal is not achieved and **false** is returned.



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

Matching would take Place as Follows – [ a -> a ], [ \_ -> X ], [ c -> Y ] & [d -> \_ ]

Therefore, Y = c.

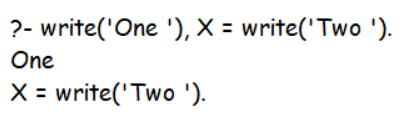


[Note that both ‘**\_’ {Anonymous Variables}** are Different. Therefore, \_ = d & \_ = X doesn’t implies X = d]

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

Here, write(‘One’) writes to the Console and is true statement.

It being followed by a comma ‘,’ -> **AND** operator which is followed by another correct statement, hence written within the console too.



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

**PROLOG Code**

*/\* Facts \*/*

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)*.*

*/\* Rules \*/*

*% Father*

father(X, Y)*:-*

    male(X),

    parent(X, Y)*.*

*% Mother*

mother(X, Y)*:-*

    female(X),

    parent(X, Y)*.*

*% Sister*

sister(X, Y)*:-*

    female(X),

    father(F, X),

    father(F, Y),

    X \= Y*.*

sister(X, Y)*:-*

    female(X),

    mother(M, X),

    mother(M, Y),

    X \= Y*.*

*% Grandmother*

grandmother(X, Y)*:-*

    female(X),

    parent(X, Z),

    parent(Z, Y)*.*

*% Cousin*

sibling(X, Y)*:-*

    father(Z, X),

    father(Z, Y),

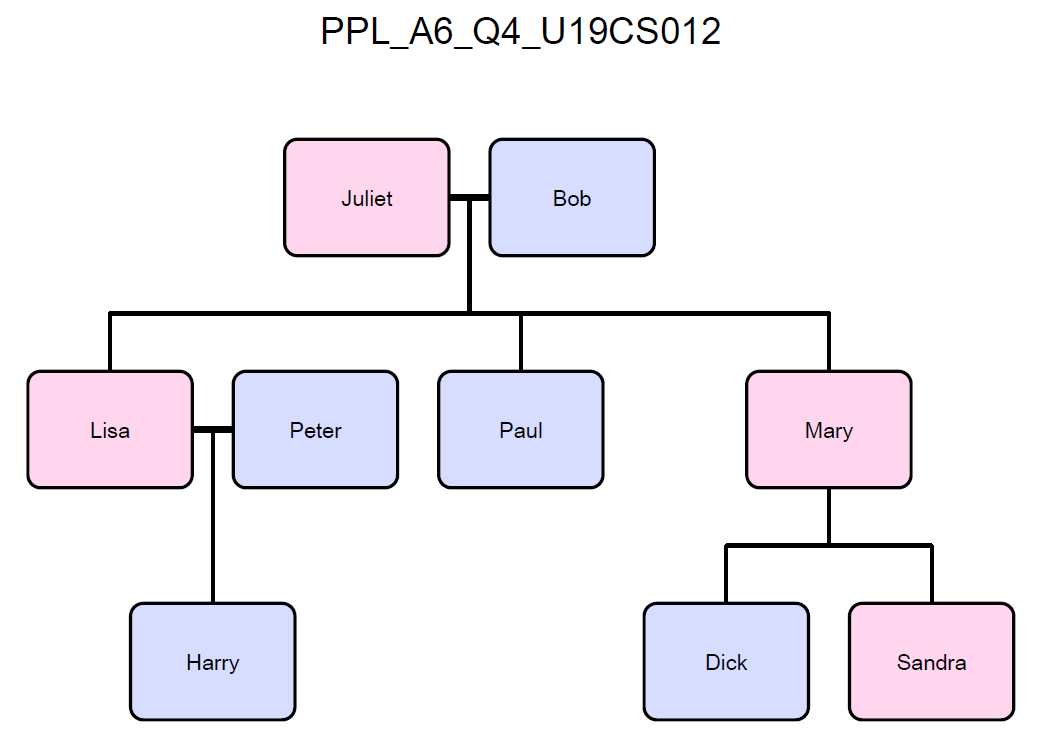
    X \= Y*.*

cousin(X, Y)*:-*

    mother(Z, X),

    mother(W, Y),

    sibling(Z, W)*.*



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

*% Father*

father(X, Y)*:-*

    male(X),

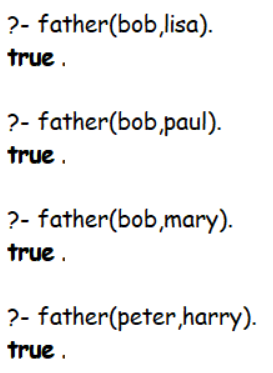
    parent(X, Y)*.*

*% Mother*

mother(X, Y)*:-*

    female(X),

    parent(X, Y)*.*



(b) sister

*% Sister*

sister(X, Y)*:-*

    female(X),

    father(F, X),

    father(F, Y),

    X \= Y*.*

sister(X, Y)*:-*

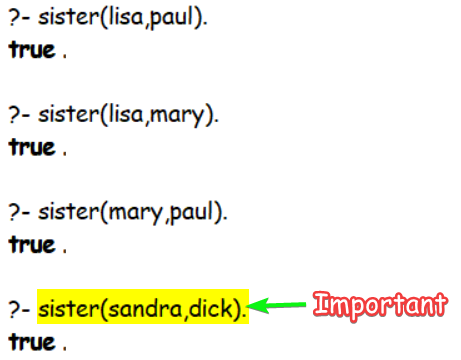
    female(X),

    mother(M, X),

    mother(M, Y),

    X \= Y*.*

Second Definition is needed for that Fourth Test Case as shown below.



(c) grandmother

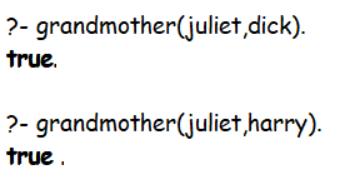
*% Grandmother*

grandmother(X, Y)*:-*

    female(X),

    parent(X, Z),

    parent(Z, Y)*.*



(d) cousin

*% Cousin*

sibling(X, Y)*:-*

    father(Z, X),

    father(Z, Y),

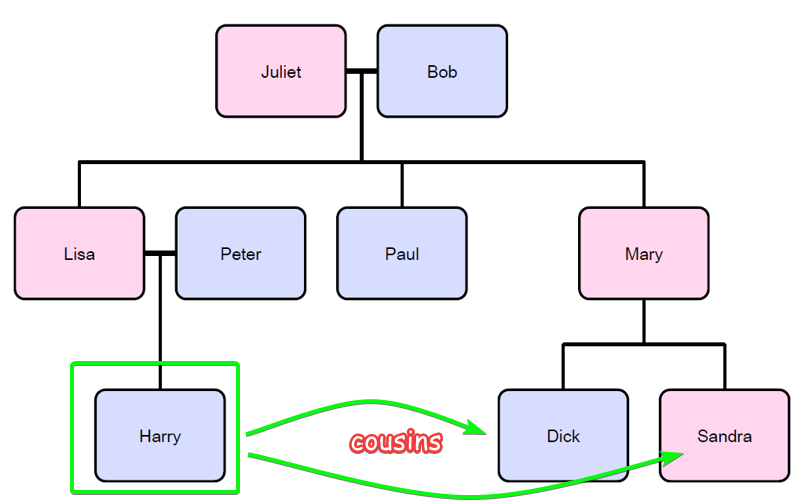
    X \= Y*.*

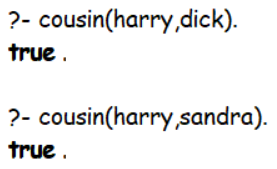
cousin(X, Y)*:-*

    mother(Z, X),

    mother(W, Y),

    sibling(Z, W)*.*





**SUBMITTED BY**: U19CS012

BHAGYA VINOD RANA