Principles of Programming Language (CS302)

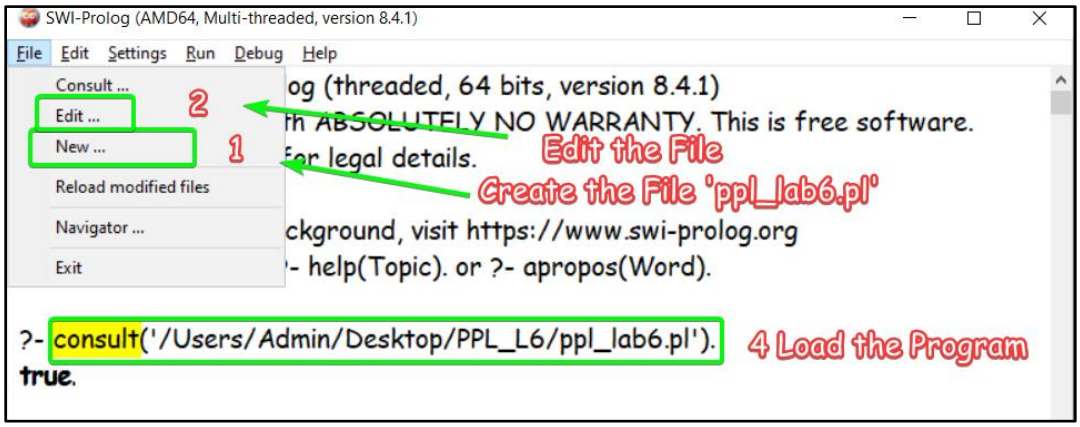
Assignment - 6

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.

ppl_lab6.pl

```
main :-
    write('Enter the PPL Assignment No - '),
    nl,
    read(Number),
    printit(Number).

printit(Number):-
    write('You have Entered '),
    write(Number),
    write(' As Assignment Number!'),
    nl.
```

```
?- main.
Enter the PPL Assignment No -
|: 6.
You have Entered 6 As Assignment Number!
true.
```

(c) Output predicates

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

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

```
?- write('Prolog in PPL'), nl, write('By Bhagya Rana').
Prolog in PPL
By Bhagya Rana
true.
```

(d) Utility Functions

halt -> causes the Prolog system to terminate

```
?- halt.
```

Window Closes after Running the Command.

statistics -> prints system statistics.

```
?- statistics.
% Started at Mon Feb 28 01:32:48 2022
% 0.219 seconds cpu time for 258,428 inferences
% 6,107 atoms, 4,472 functors, 3,085 predicates, 43 modules, 123,477 VM-codes
%
%                    Limit   Allocated      In use
% Local  stack:        -      20 Kb    2,224  b
% Global stack:        -     124 Kb       67 Kb
% Trail  stack:        -      30 Kb    1,408  b
%        Total:    1,024 Mb     174 Kb       71 Kb
%
% 2 garbage collections gained 166,800 bytes in 0.000 seconds.
% 4 atom garbage collections gained 729 atoms in 0.000 seconds.
% 7 clause garbage collections gained 149 clauses in 0.000 seconds.
% Stack shifts: 2 local, 1 global, 2 trail in 0.000 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?

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

atom

An atom, in Prolog, means a single data item. It may be of one of four types:

- a string atom, like 'This is a string' or
- a symbol, like `likes`, `john`, and `pizza`, in `likes(john, pizza)`. Atoms of this type must start with a lower case letter. They can include digits (after the initial lower-case letter) and the underscore character (_).
- the empty list `[]`. This is a strange one: other lists are not atoms. If you think of an atom as something that is not divisible into parts (the original meaning of the word *atom*, though subverted by subatomic physics) then `[]` being an atom is less surprsing, since it is certainly not divisible into parts.
- strings of special characters, like `<--->`, `...`, `===>`. When using atoms of this type, some care is needed to avoid using strings of special characters with a predefined meaning, like the neck symbol `:-`, the cut symbol `!`, and various arithmetic and comparison operators.
  The available special characters for constructing this class of atom are: `+, -, *, /, <, >, =, :, ., &, _`, and ~.

Numbers, in Prolog, are not considered to be atoms.

Atom	Valid	Reason
f	✓	Not Beginning with a Capital Letter
loves(john,mary)	✗	Clauses are not Atomic.
Mary	✗	Should Start with Small Case Letter
_c1	✗	Since it is Variable
'Hello'	✓	String Atom
this_is_it	✓	Starts with Lower Case Letter

```
?- 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 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?

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

```
| ?- foo(a,b) = foo(X,Y).        ** Two complex terms unify if they are **
     X=a                         ** of the same arity, have the same principal**
     Y=b                         ** functor and their arguments unify **
yes
| ?- foo(a,Y) = foo(X,b).        ** Instantiation of variables may occur **
     Y=b                         ** in either of the terms to be unified **
     X=a
yes
| ?- foo(a,b) = foo(X,X).        ** In this case there is no unification **
no                               ** because foo(X,X)  must have the same **
                                 ** 1st and 2nd arguments **
```

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

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

Why? Since **John** is Variable that is Unified with Value mary and **Mary** is 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

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

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

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

?- 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 <u>not achieved</u> and **false** is returned.

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

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

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

Therefore, Y = c.

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

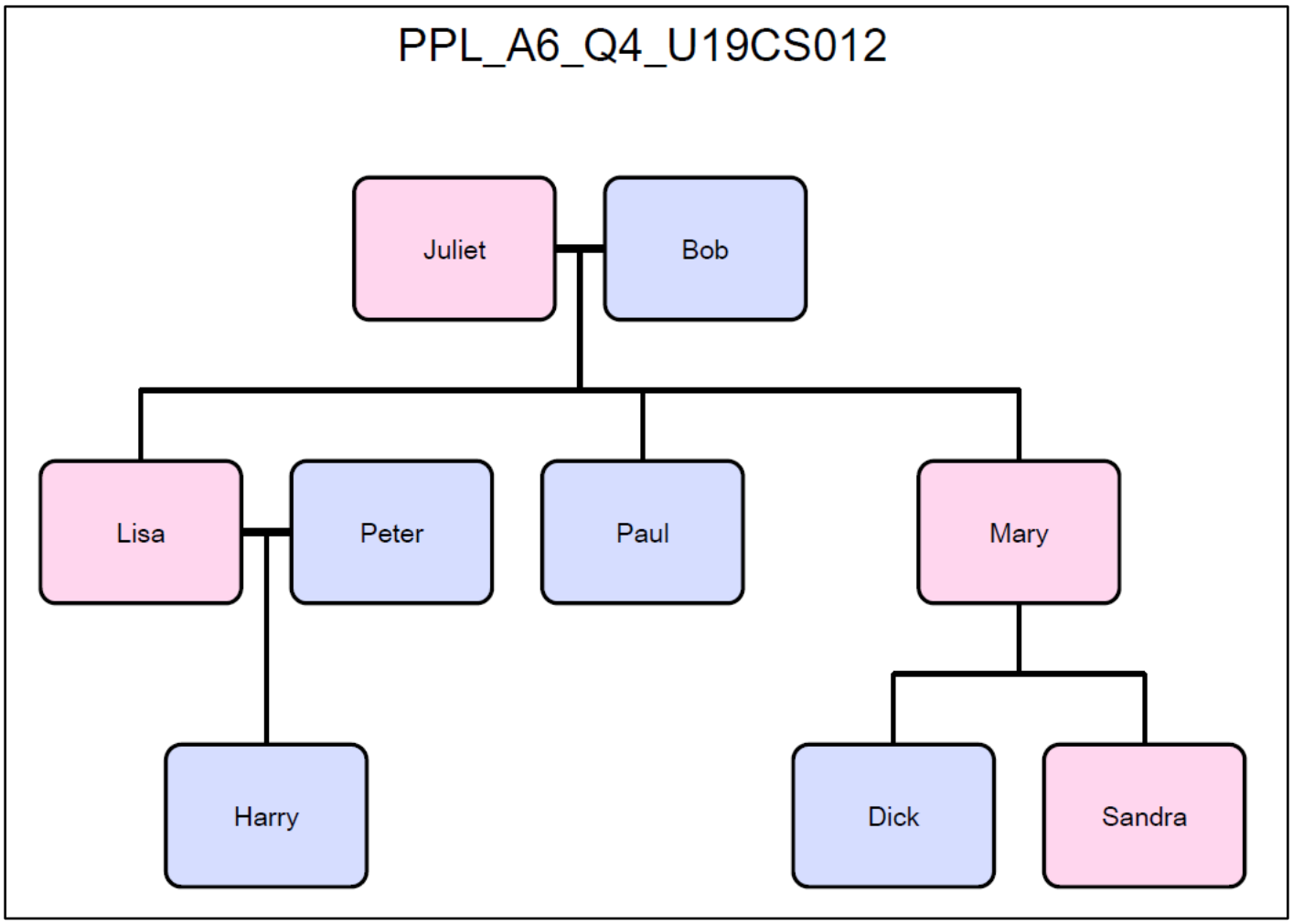
```
?- write('One '), X = write('Two ').
One
X = write('Two ').
```

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).
```

```
?- father(bob,lisa).
true .

?- father(bob,paul).
true .

?- father(bob,mary).
true .

?- father(peter,harry).
true .
```

(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 <u>Fourth Test Case</u> as shown below.

```
?- sister(lisa,paul).
true .

?- sister(lisa,mary).
true .

?- sister(mary,paul).
true .

?- sister(sandra,dick).
true .
```

Important

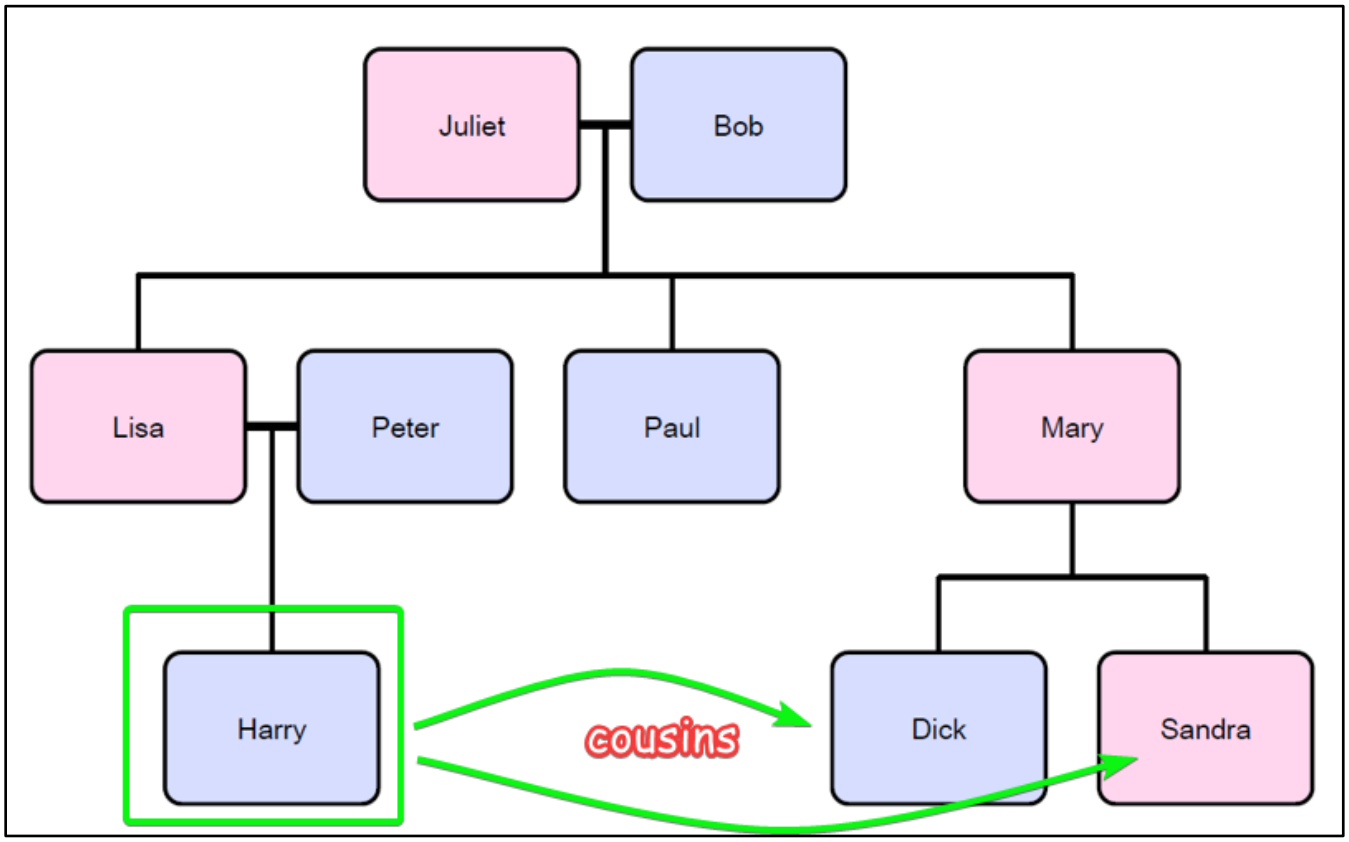
(c) grandmother

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

```
?- grandmother(juliet,dick).
true.

?- grandmother(juliet,harry).
true .
```

(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).
```




```
?- cousin(harry,dick).
true .

?- cousin(harry,sandra).
true .
```

SUBMITTED BY: U19CS012

BHAGYA VINOD RANA