**PRACTICAL - 1**

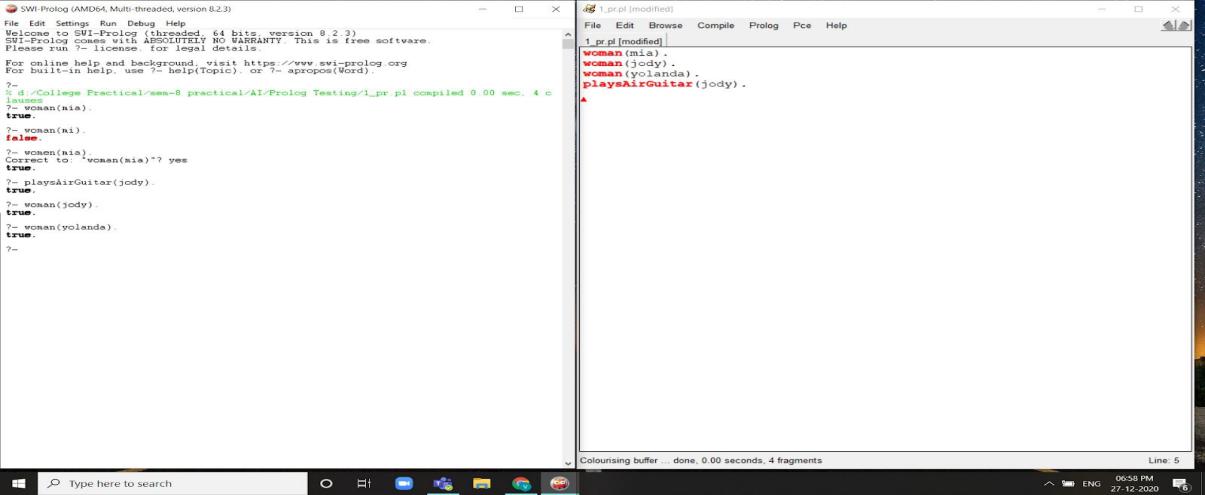
# AIM :

* 1. **Write a prolog program to implement different kinds of knowledge bases.**

1. **Knowledge Base 1 : Program :**

woman(mia).

woman(jody). woman(yolanda). playsAirGuitar(jody).



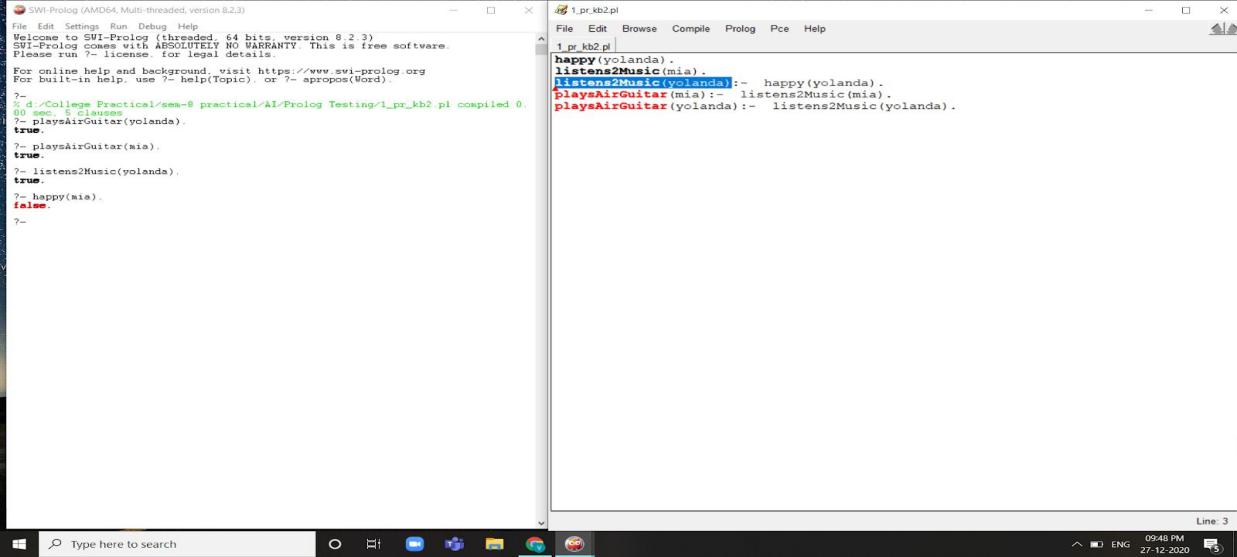
# Fig. 1.1 Knowbase 1

1. **Knowledge Base 2 : Program :**

happy(yolanda).

listens2Music(mia). listens2Music(yolanda):- happy(yolanda). playsAirGuitar(mia):- listens2Music(mia).

playsAirGuitar(yolanda):- listens2Music(yolanda).



# Fig 1.2 Knowbase 2

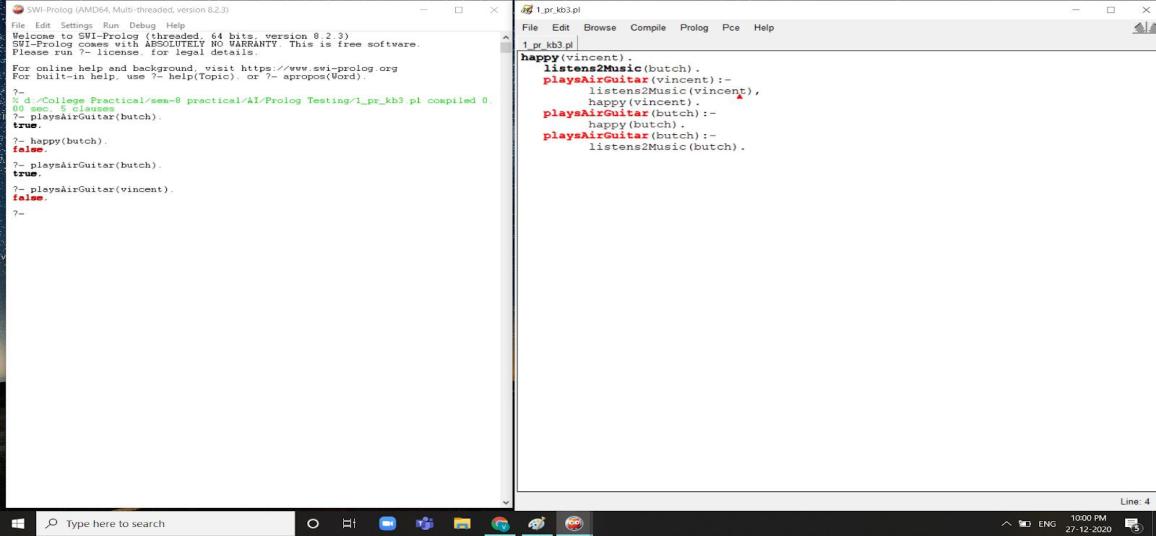
1. **Knowledge Base 3 : Program :**

happy(vincent). listens2Music(vincent). playsAirGuitar(vincent):-

listens2Music(vincent), happy(vincent).

playsAirGuitar(butch):- happy(butch).

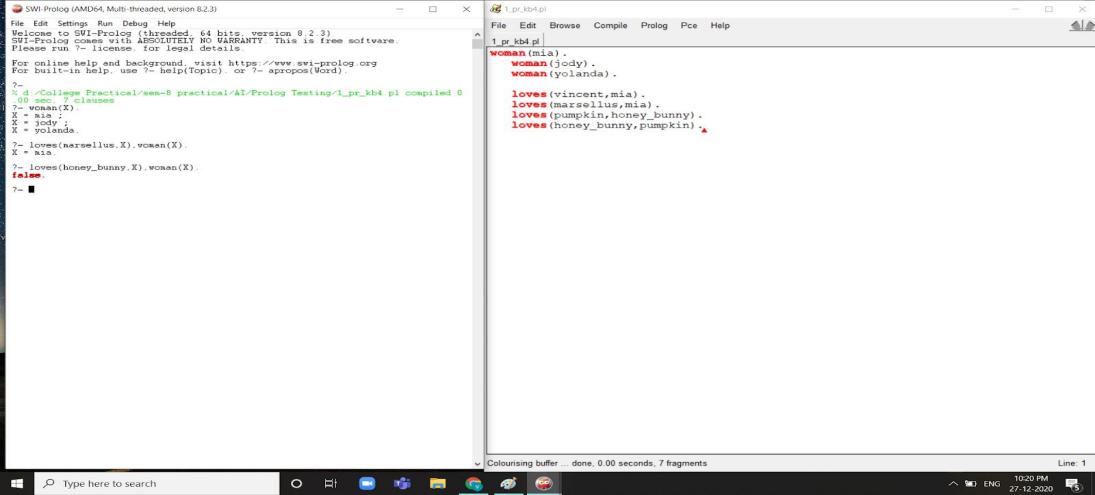
playsAirGuitar(butch):- listens2Music(butch).



# Fig 1.3 Knowbase 3

1. **Knowledge Base 4 : Program :**

woman(mia). woman(jody). woman(yolanda).

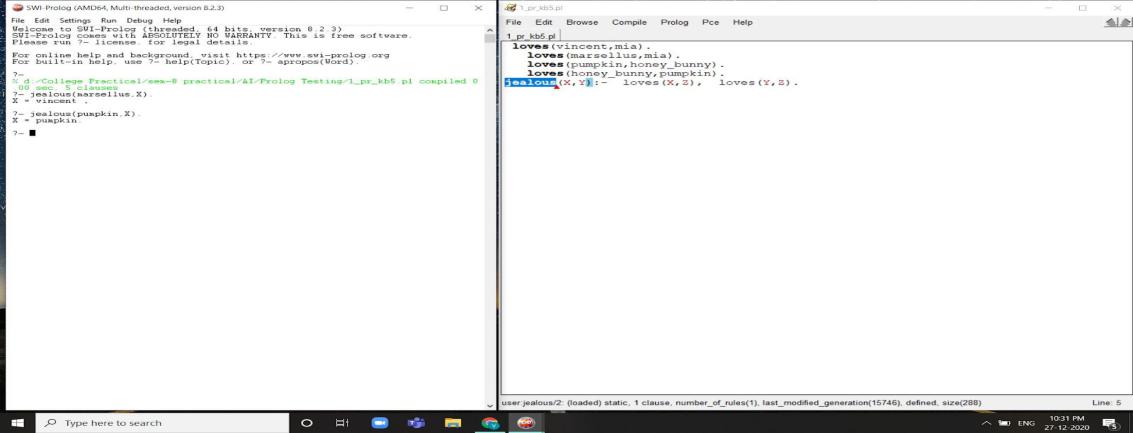
loves(vincent,mia). loves(marsellus,mia). loves(pumpkin,honey\_bunny). loves(honey\_bunny,pumpkin).

# Fig 1.4 Knowbase 4

1. **Knowledge Base 5 : Program :**

loves(vincent,mia).

loves(marsellus,mia). loves(pumpkin,honey\_bunny). loves(honey\_bunny,pumpkin).

jealous(X,Y):- loves(X,Z), loves(Y,Z).

# Fig 1.5 Knowbase 5

* 1. **Write a program which contains three predicates: male, female, parent. Make rules for following family relations: father, mother, grandfather, grandmother, brother, sister, uncle, aunt, nephew and niece.**

**Program :**

male(ramji). male(amu). male(mahesh). female(savita). female(jignasa). female(asha). female(veni). female(mitu). female(riya). female(mahek). parent(amu,veni). parent(jignasa,veni). parent(amu,mitu). parent(jignasa,mitu). parent(mahesh,riya). parent(asha,riya). parent(mahesh,mahek). parent(asha,mahek). parent(ramji,amu). parent(savita,amu). parent(ramji,mahesh). parent(savita,mahesh).

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

daughter(X,Y):-female(X),parent(Y,X).

mother(X,Y):-parent(X,Y),female(X).

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

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

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

grandfather(X,Y):-parent(X,Z),parent(Z,Y),male(X),male(Z).

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

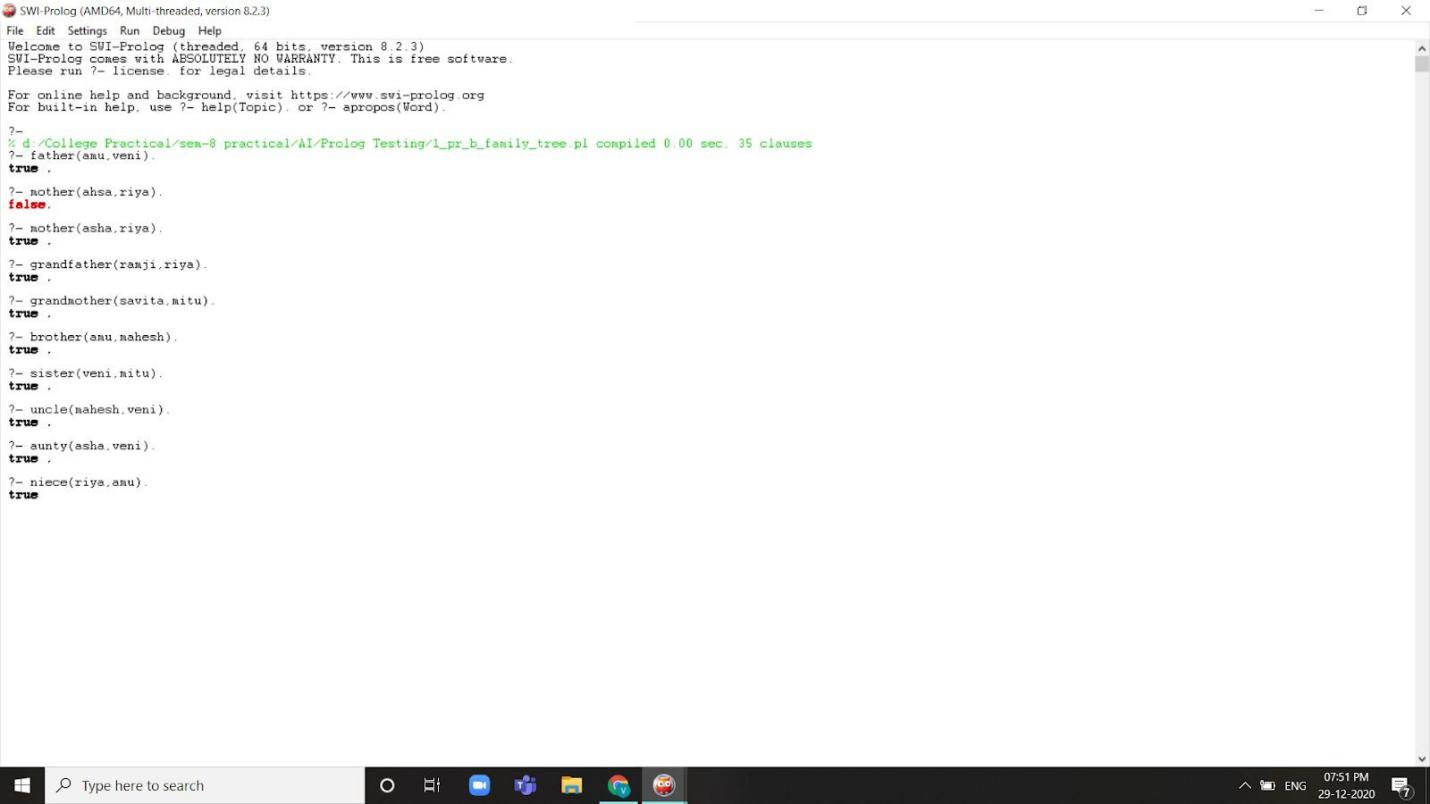
uncle(X,Y):-parent(Z,Y),brother(Z,X),male(X).

wife(X,Y):-female(X),male(Y),parent(X,Z),parent(Y,Z).

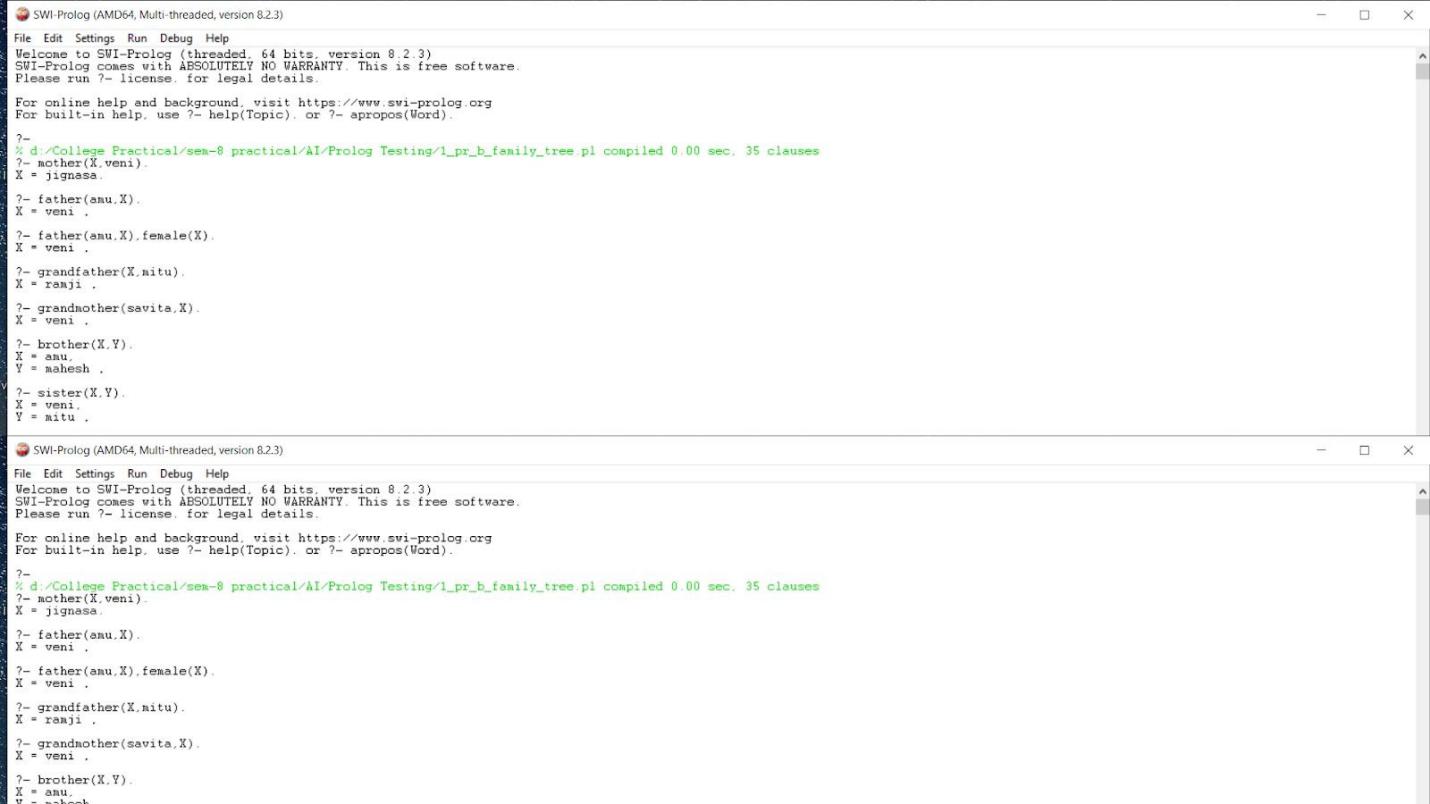
aunty(X,Y):-female(X),parent(Z,Y),brother(P,Z),wife(X,P).

nephew(X,Y):-male(X),son(X,Z),brother(Z,Y).

niece(X,Y):-female(X),daughter(X,Z),brother(Z,Y).



# Fig 1.6 Family Tree 1



**Fig 1.7 Family Tree 2**

**PRACTICAL – 2**

1. **Write a PROLOG program to implement Tower Of Hanoi Problem**

hanoi:- hanoi(5).

hanoi(N):- moveit(N,1,2,3).

moveit(1,X,Y,\_) :-

write('Move top disk from '),

write(X), write(' to '),

write(Y), nl.

moveit(N,X,Y,Z) :-

N > 1,

M is N-1,

moveit(M,X,Z,Y),

moveit(1,X,Y,\_),

moveit(M,Z,Y,X).

**OUTPUT :**

?- hanoi(3).

Move top disk from 1 to 2

Move top disk from 1 to 3

Move top disk from 2 to 3

Move top disk from 1 to 2

Move top disk from 3 to 1

Move top disk from 3 to 2

Move top disk from 1 to 2

true.

1. **Write a program to solve Water-Jug Problem**

jug(2, \_).

jug(0, 2):- write('(0, 2)'), nl,

write('(2, 0)'), nl.

jug(4, 0):- write('(4, 0)'), nl,

jug(0, 0).

jug(4, 3):- write('(4, 3)'), nl,

jug(0, 0).

jug(3, 0):- write('(3, 0)'), nl,

jug(3, 3).

jug(X, 0):- write('('),write(X), write(', 0)'), nl,

jug(0, 3).

jug(0, 3):- write('(0, 3)'), nl,

jug(3, 0).

jug(0, X):- write('(0, '),write(X),write(')'), nl,

jug(0, 0).

jug(3, 3):- write('(3, 3)'), nl,

jug(4, 2).

jug(4, 2):- write('(4, 2)'), nl,

write('(2, 0)'), nl,

jug(2, 0).

jug(X, Y):- X>4, fail,

Y>3, fail.

**OUTPUT:**

?- jug(4,3).

(4, 3)

(0, 0)

(0, 3)

(3, 0)

(3, 3)

(4, 2)

(2, 0)

true.

**PRACTICAL – 3**

1. **Write a program to input user name and password from user and repeatedly asking if any one of them is wrong.**

chk(X,Y):-

X = user1, Y = pass1 ;

X = user2, Y = pass2.

login:- repeat,

write('Enter username : '),

read(U),

write('Enter password : '),

read(P),

chk(U,P),

write('Correct user').

**OUTPUT:**

?- login.

Enter username : user1.

Enter password : pass2.

Enter username : user2.

Enter password : pass1.

Enter username : user2.

Enter password : pass2.

Correct user

true.

1. **Write a program that list four addresses in a label form, each address should list a name, one-line address, city, state & pin-code.**

printX :-

write('ABC'), nl, write('E-1-43'), nl, write('Rajkot'), nl, write('Gujarat - '),

write('360001'), nl, nl,

write('DEF'), nl, write('5C-738'), nl, write('Gandhinagar'), nl, write('Gujarat - '),

write('381006'), nl, nl,

write('PQR'), nl, write('C-13'), nl, write('Jamnagar'), nl, write('Gujarat - '),

write('361008'), nl, nl,

write('XYZ'), nl, write('S-54'), nl, write('Ahmedabad'), nl, write('Gujarat - '),

write('380001'), nl, nl.

**OUTPUT:**

?- printX.

ABC

E-1-43

Rajkot

Gujarat - 360001

DEF

5C-738

Gandhinagar

Gujarat - 381006

PQR

C-13

Jamnagar

Gujarat - 361008

XYZ

S-54

Ahmedabad

Gujarat - 380001

true.

**PRACTICAL – 4**

**Convert the given PROLOG predicates into Semantic Net. cat(tom). cat(cat1). mat(mat1). sat\_on(cat1,mat1). bird(bird1). caught(tom,bird1). like(X,cream) :– cat(X). mammal(X) :– cat(X). has(X,fur) :– mammal(X). animal(X) :– mammal(X). animal(X) :– bird(X). owns(john,tom). is\_coloured(tom,ginger).**

cat(tom).

cat(cat1).

mat(mat1).

sat\_on(cat1,mat1).

bird(bird1).

caught(tom,bird1).

like(X,cream) :– cat(X).

mammal(X) :– cat(X).

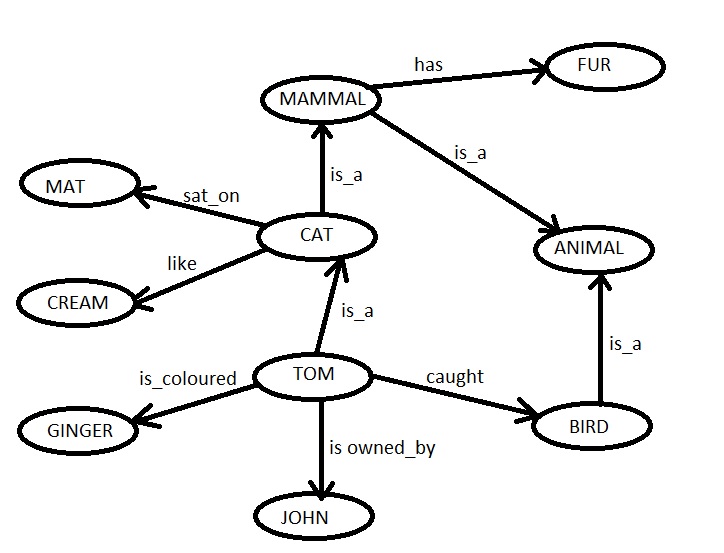
has(X,fur) :– mammal(X).

animal(X) :– mammal(X).

animal(X) :– bird(X).

owns(john,tom).

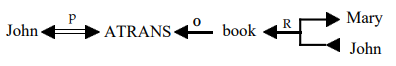
is\_coloured(tom,ginger).



**PRACTICAL – 5**

**Write the Conceptual Dependency for the given statements.**

**(a) John gives Mary a book**



**(b)John gave Mary the book yesterday.**

Yesterday

