ASSIGNMENT # 3:

ASSIGNMENTS

COURSE NAME: ARTIFICIAL INTELLIGENCE (LAB)

CLASS ID:103783

NAME: JAWERIA ASIF

INSTRUCTOR: SIR MINHAL

DUE DATE: 4-FEBRARY-2020

DUE DAY: TUESDAY

SUBMITTED TO: SIR MINHAL

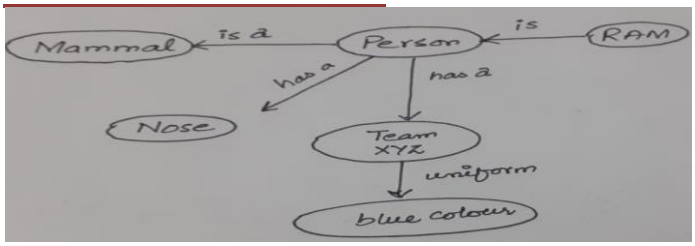
QUESTION # 01:

A mammal is a class, a person shares the properties of the mammal, "Ram" is the instance of person who is the team member of team named XYZ having uniform of color blue. Person has a property that it has nose.

CODE:

```
person(mammal).
nose(person).
ram(person).
member(ram,team).
team(xyz,blueuniform).
```

SEMANTIC DIAGRAM



OUTPUT:

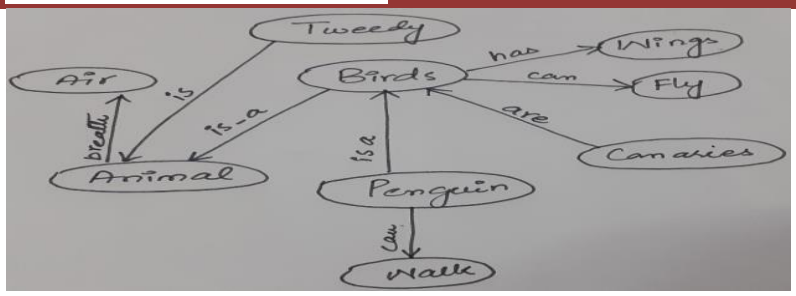
QUESTION # 02:

b) A bird has wings and can fly, since birds have wings and can fly, and canaries is also a type of birds, it seems reasonable that they also have wings and can fly. A penguin is also a bird and they can travel through walk. A bird is animal, Tweedy is also an animal, and Animals breathes in Air.

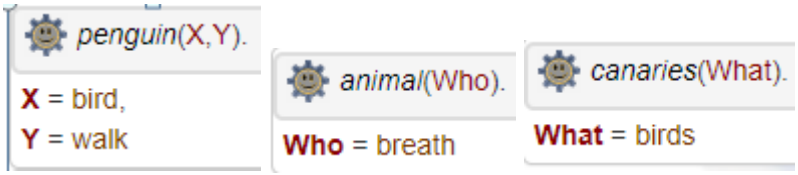
CODE:

```
canaries(birds).
animal(breath).
birds(animal).
birds(wings,fly).
penguin(bird,walk).
tweedy(animal).
```

SEMANTIC DIAGRAM



OUTPUT:



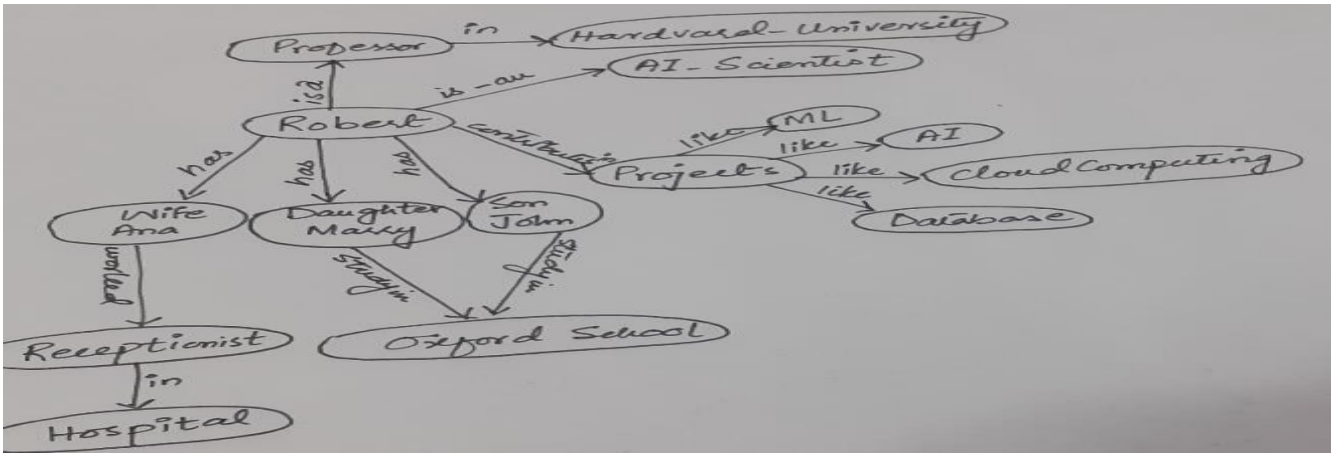
QUESTION # 03:

Robert is a former AI Scientist who works as a professor in Harvard University. He contributes in many projects like Web Development, AI, ML, Cloud Computing and Databases; He has a family with a wife named Ana who works as a Receptionist in a hospital, His 2 children named are John and Marry who study in a school which is near their home. The School named is Oxford.

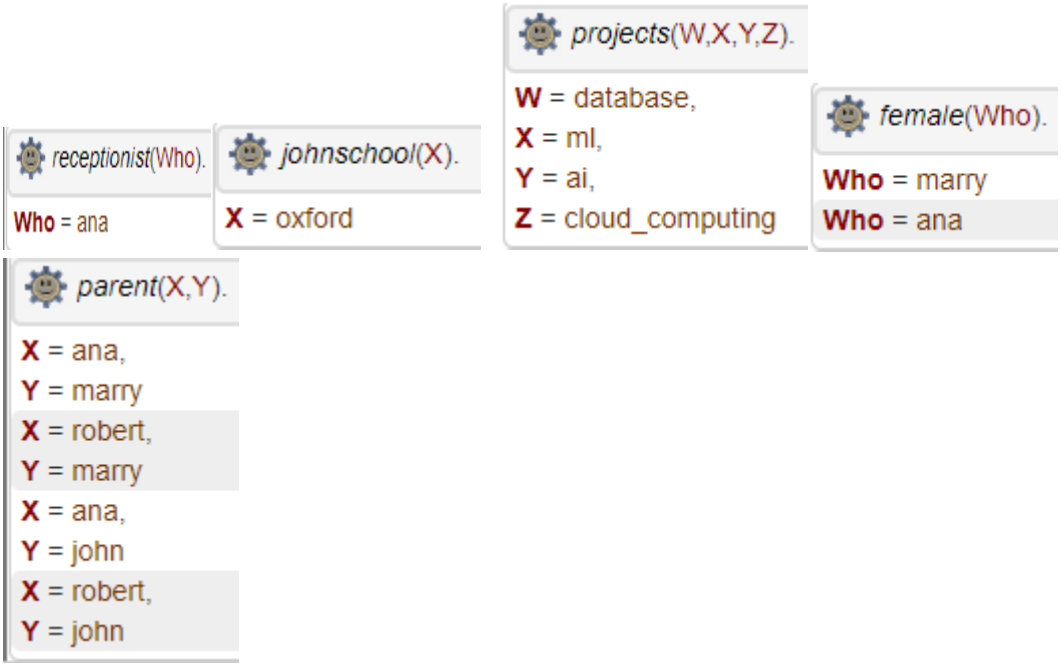
CODE:

```
male(robert).
male(john).
female(marry).
female(ana).
robert(ai_scientist).
projects(database,ml,ai,cloud_computing).
contribute(robert,projects).
professor(robert,hardvarduniversity).
receptionist(ana).
parent(ana,marry).
parent(robert,marry).
parent(ana,john).
parent(robert,john).
child(X,Y):- parent(Y,X).
son(X,Y):- male(X),child(X,Y).
daughter(X,Y):- female(X),child(X,Y).
marryschool(oxford).
johnschool(oxford).
```

SEMANTIC DIAGRAM:



OUTPUT:


receptionist(Who).
Who = ana
johnschool(X).
X = oxford
projects(W,X,Y,Z).
W = database,
X = ml,
Y = ai,
Z = cloud_computing
female(Who).
Who = marry
Who = ana
parent(X,Y).
X = ana,
Y = marry
X = robert,
Y = marry
X = ana,
Y = john
X = robert,
Y = john

QUESTION # 04:

MAP COLOURING PROBLEM CSP:

```
adj(1,5).
adj(5,1).
adj(4,5).
adj(5,4).
adj(2,4).
adj(4,2).
adj(1,4).
adj(4,1).
adj(2,1).
adj(1,2).
adj(3,2).
adj(2,3).
adj(4,3).
adj(3,4).
adj(1,3).
adj(3,1).
color(1, red, a).
color(2, blue, a).
color(3, green, a).
color(4, yellow, a).
color(5, blue, a).
color(1, red, b).
color(2, blue, b).
color(3, green, b).
color(4, yellow, b).
color(5, blue, b).
conflict(scheme):-
    adj(X,Y),color(X,coloring,Scheme),color(Y,coloring,Scheme).
```

QUESTION # 05:

INTRODUCTION TO OPERATORS:

```
a is_parent_of b.
a is_parent_of c.
a is_parent_of d.
b is_parent_of e.
b is_parent_of f.
c is_parent_of g.
c is_parent_of h.
c is_parent_of i.
d is_parent_of j.
j is_parent_of q.
j is_parent_of r.
j is_parent_of s.
i is_parent_of o.
i is_parent_of p.
h is_parent_of n.
f is_parent_of l.
f is_parent_of m.
m is_parent_of t.
n is_parent_of u.
n is_parent_of v.
:-op(500,xfx,'is_parent_of').
:-op(500,xfx,'is_sibling_of').
:-op(500,xfx,'is_at_same_level').
X is_sibling_of Y:- Z is_parent_of X, Z is_parent_of Y, X\==Y.
leaf_node(Node):-\+ is_parent_of(Node,Child).
X is_at_same level X.
X is_at_same level Y:- W is_parent_of X, Z is_parent_of Y, W is_at_same_level Z.
path(a).
path(Node):- X is_parent_of Node, path(X), write(X), write('-').
locate(Node):-path(Node), write(Node),nl.
```

OUTPUT:

```
a is_parent_of b.
true
b is_at_same_level c.
true
```

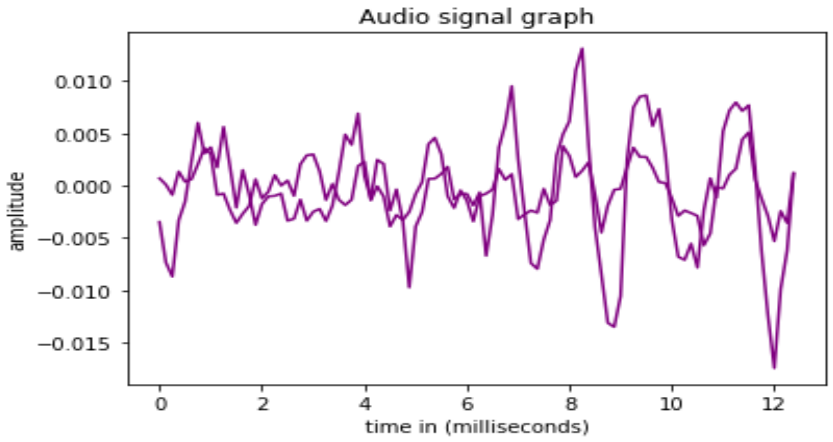
QUESTION #06:

PYTHON CODE FOR GENERATING AUDIO SIGNAL GRAPH

```
In [9]: import numpy as np
        from matplotlib import pyplot as plt
        from scipy.io import wavfile
        freq,signal=wavfile.read("file_example_WAV_1MG.wav")
        print("Signal datatype:  ",signal.dtype)
        print("Signal duration:  ",round(signal.shape[0]/float(freq),2),'seconds')
        signal=signal/np.power(2,15)
        signal=signal[:100]
        time=1000*np.arange(0,len(signal),1)/float(freq)
        plt.plot(time,signal,color="purple")
        plt.xlabel('time in (milliseconds)')
        plt.ylabel('amplitude')
        plt.title('Audio signal graph')
        plt.show()
```

OUTPUT:

```
Signal datatype:   int16
Signal duration:   33.53 seconds
```



QUESTION #07:

Speech Recognition

In [13]:
```
import  speech_recognition  as  sr
r = sr.Recognizer()
with sr.Microphone() as source:
    print('Speak Anything...')
    audio=r.listen(source)
    try:
        text=r.recognize_google(audio)
        print('You said: {}'.format(text))
    except:
        print('Sorry we could not understand you spoken words')
```

OUTPUT:

```
Speak Anything :
You said : this is speech recognition program using python

In [6]:
```

QUESTION # 08:

1. Interview your family members and construct a family tree in Prolog Language containing the predicate and rules. Switch roles and perspectives as necessary to perform or answer the following: Use the following predicate and make the rules from them. Use or Assume necessary information when and where needed

Predicates/Facts

a) Male b) Female c) Parent d) Married

Rules a) Mother b) Father c) Brother d) Sister e) Aunt f) Uncle g) Siblings h) Grandmother i) Grandfather j) Cousin k) Husband l) Wife m) Son n) Daughter o) Has Child

SOURCE CODE FOR FAMILY TREE IN PROLOG

```
male(rais).
male(amin).
male(asif).
male(arif).
male(amir).
male(tayyab).
male(zaki).
male(ashir).
male(sufyan).
male(shams).

female(saida).
female(hajra).
female(najma).
female(salma).
female(farhat).
female(sehrish).
female(jaweria).
female(mahnoor).
female(hoorain).
female(noshaba).
female(rukha).

parent_of(rais,asif).
parent_of(rais,arif).
parent_of(rais,amir).
parent_of(saida,asif).
parent_of(saida,arif).
parent_of(saida,amir).

parent_of(amin,najma).
parent_of(hajra,najma).
parent_of(amin,salma).
parent_of(hajra,salma).

parent_of(salma,rukha).
parent_of(shams,rukha).
```

```
parent_of(salma,sufyan).
parent_of(shams,sufyan).

parent_of(asif,jaweria).
parent_of(asif,tayyab).
parent_of(najma,jaweria).
parent_of(najma,tayyab).

parent_of(arif,mahnoor).
parent_of(arif,hoorain).
parent_of(arif,zaki).
parent_of(farhat,mahnoor).
parent_of(farhat,hoorain).
parent_of(farhat,zaki).

parent_of(amir,ashir).
parent_of(amir,noshaba).
parent_of(sehrish,ashir).
parent_of(sehrish,noshaba).

grandmother(X,Y) :- mother(X,P), parent_of(P,Y).
grandfather(X,Y) :- father(X,P), parent_of(P,Y).
grandson(X,Y) :- son(X,P), parent_of(Y,P).
granddaughter(X,Y) :- daughter(X,P), parent_of(Y,P).
aunt(X,Y) :- sister(X,P), parent_of(P,Y).
aunt(X,Y) :- wife(X,P), sibling(P,Q), parent_of(Q,Y).
uncle(X,Y) :- brother(X,P), parent_of(P,Y).
uncle(X,Y) :- husband(X,P), sibling(P,Q), parent_of(Q,Y).
niece(X,Y) :- daughter(X,P), sibling(P,Y).
niece(X,Y) :- daughter(X,P), sibling(P,Q), spouse(Q,Y).
nephew(X,Y) :- son(X,P), sibling(P,Y).
nephew(X,Y) :- son(X,P), sibling(P,Q), spouse(Q,Y).
cousin(X,Y) :- parent_of(P,X), sibling(P,Q), parent_of(Q,Y).
child(X,Y) :- parent_of(Y,X).
spouse(X,Y) :- child(P,X), child(P,Y).
husband(X,Y) :- male(X), spouse(X,Y).
wife(X,Y) :- female(X), spouse(X,Y).
son(X,Y) :- male(X), child(X,Y).
daughter(X,Y) :- female(X), child(X,Y).
mother(X,Y) :- female(X), parent_of(X,Y).
father(X,Y) :- male(X), parent_of(X,Y).
sibling(X,Y) :- parent_of(P,X), parent_of(P,Y), X\=Y.
brother(X,Y) :- male(X), sibling(X,Y).
sister(X,Y) :- female(X), sibling(X,Y).
matGrandma(X,Y) :- mother(X,P), mother(P,Y).
patGrandma(X,Y) :- mother(X,P), father(P,Y).
matGrandpa(X,Y) :- father(X,P), mother(P,Y).
patGrandpa(X,Y) :- father(X,P), father(P,Y).
```

QUESTION # 09:

2. Assume you are the now a systems analyst on the project. As the systems analyst, you now have been tasked to do the following for the construction of semantic network:

a) A man whose profession consists in catching and selling fish moves fast on the ground in direction of...
b) Every truck is a vehicle.
Every trailer truck is a truck that has
As part a trailer,
An unloaded weight, which is a weight measure,
A maximum gross weight, which is a weight measure,
A cargo capacity, which is a volume measure,
And a number of wheels, which is the integer 18.

IMAGE OF SOLUTION #9PART A:

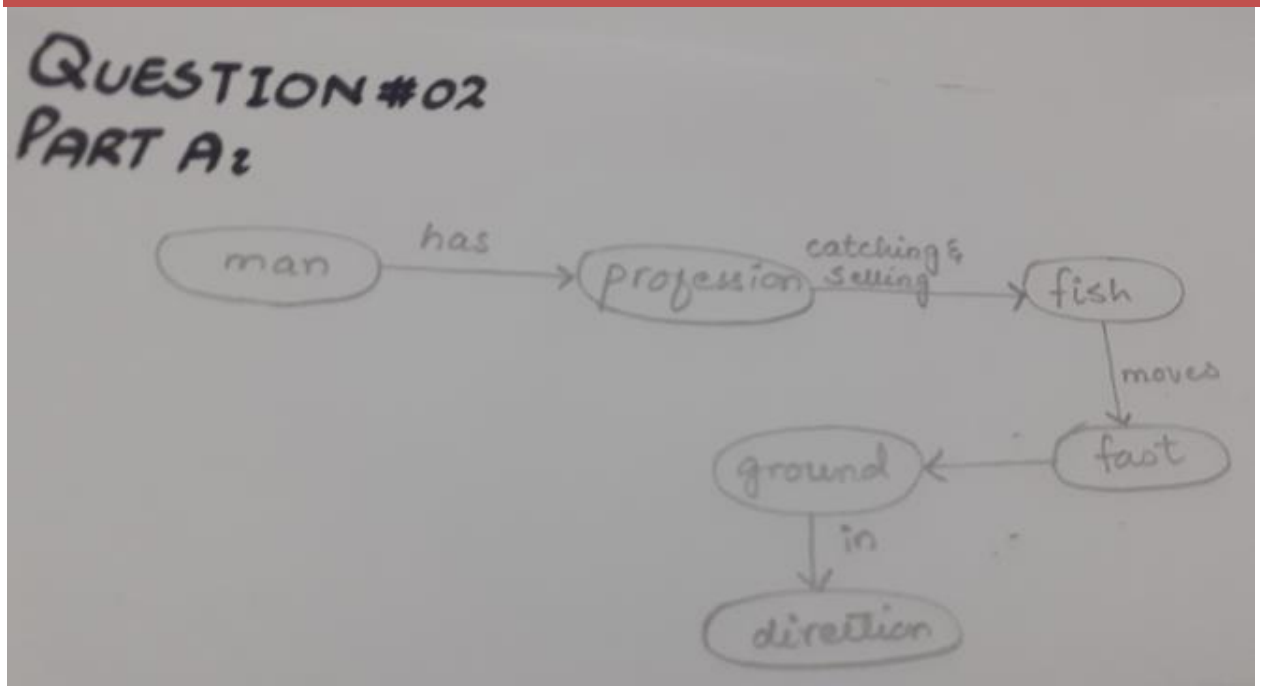
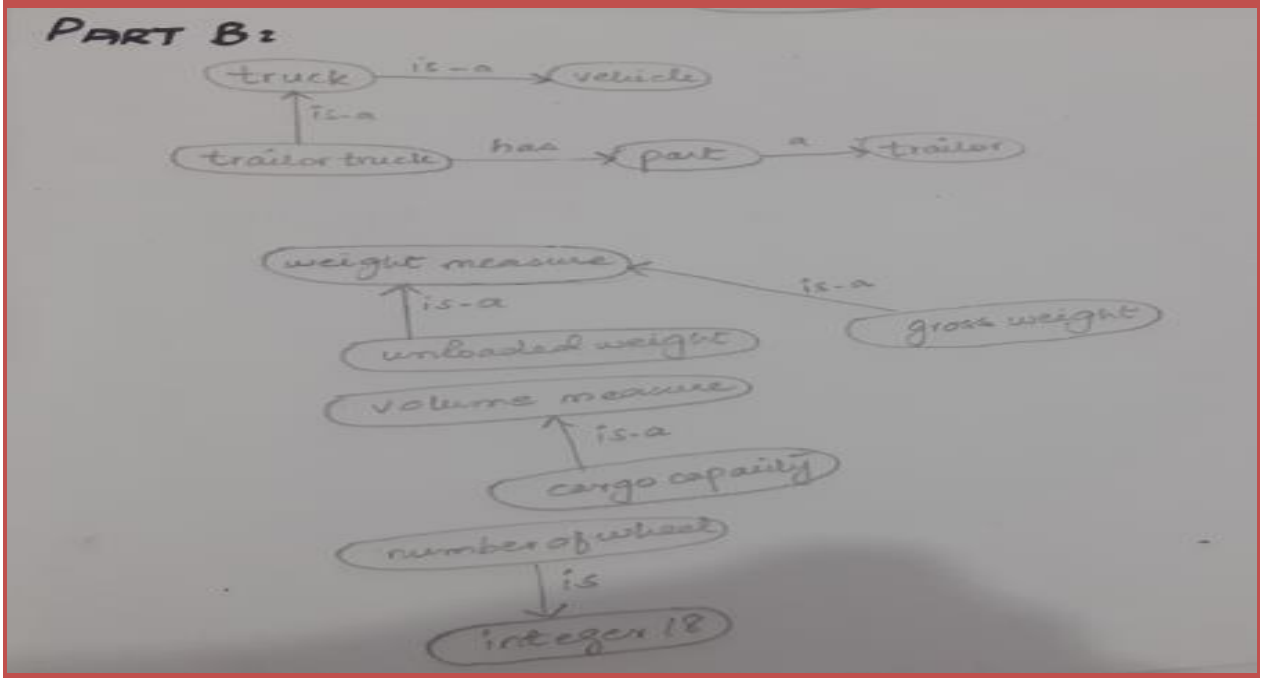


IMAGE OF SOLUTION # 9 PART B:



QUESTION # 10:

3. Transform the following Semantic Network into Human Understandable format (English) and also defined the prolog predicate/fact or rules:

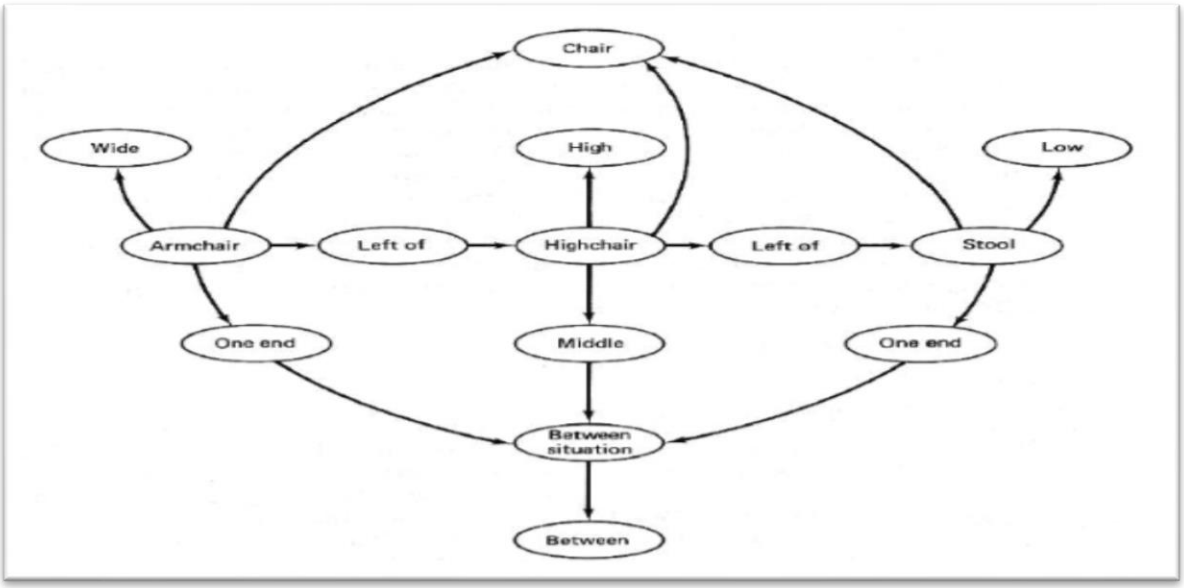
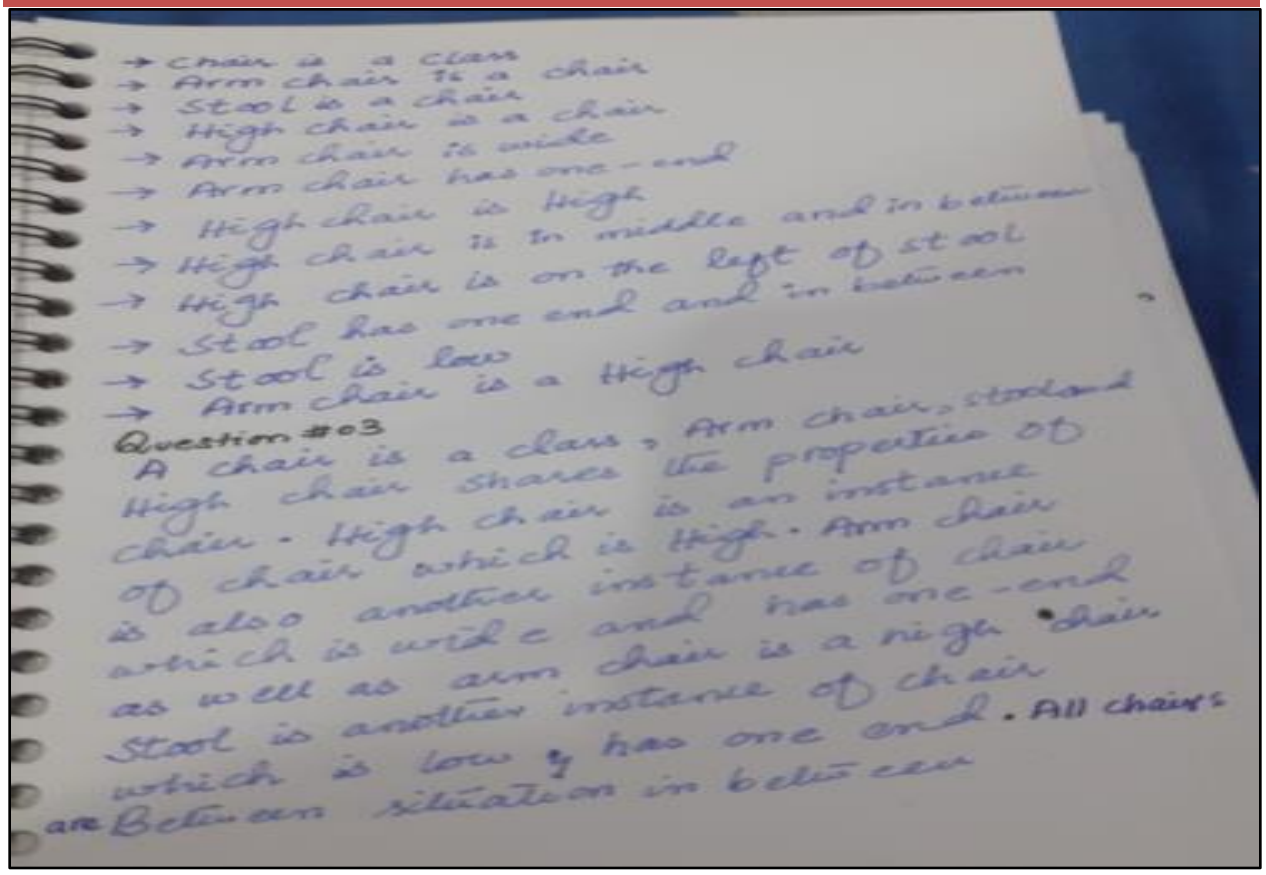


IMAGE OF SOLUTION # 10:

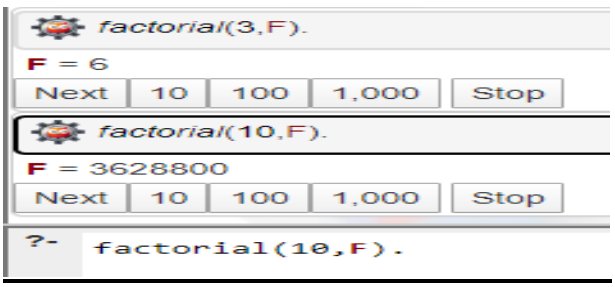


_chair(armchair).
chair(stool).
chair(high).
armchair(oneend,wide,high,left).
stool(oneend,low).
high(middle).

QUESTION # 11: FINDING FACTORIAL

```
1 factorial(0,1).
2 factorial(N,F):- N>0, N1 is N-1,factorial(N1,F1), F is N*F1.
```

OUTPUT:



QUESTION # 12: SIZE OF LINK LIST

```
size([],0).
size([H|T],N):-size(T,N1),N is N1+1.
```

OUTPUT:

```
N = 2
```

QUESTION # 13: SUM OF LIST

```
sum([],0).
sum([H|T],N):-sum(T,N1),N is N1+H.
```

QUESTION#14:

Reverse List Code:

```
reverse([],A,A).

    reverse([H|T],A,Ans):- reverse(T,A,[H|Ans]).
```

TRACE MODE IN QUERY WINDOW:

```
trace, (reverse([1,2,3,4,5,6,7,8],Prev,[])).
```

```
trace, (reverse([1,2,3,4,5,6,7,8],Prev,[])).
      Call: reverse([1, 2, 3, 4, 5, 6, 7, 8], _3498, [])
      Call: reverse([2, 3, 4, 5, 6, 7, 8], _3498, [1])
      Call: reverse([3, 4, 5, 6, 7, 8], _3498, [2, 1])
      Call: reverse([4, 5, 6, 7, 8], _3498, [3, 2, 1])
      Call: reverse([5, 6, 7, 8], _3498, [4, 3, 2, 1])
      Call: reverse([6, 7, 8], _3498, [5, 4, 3, 2, 1])
      Call: reverse([7, 8], _3498, [6, 5, 4, 3, 2, 1])
      Call: reverse([8], _3498, [7, 6, 5, 4, 3, 2, 1])
      Call: reverse([], _3498, [8, 7, 6, 5, 4, 3, 2, 1])
      Exit: reverse([], [8, 7, 6, 5, 4, 3, 2, 1], [8, 7, 6, 5, 4, 3, 2, 1])
      Exit: reverse([8], [8, 7, 6, 5, 4, 3, 2, 1], [7, 6, 5, 4, 3, 2, 1])
      Exit: reverse([7, 8], [8, 7, 6, 5, 4, 3, 2, 1], [6, 5, 4, 3, 2, 1])
      Exit: reverse([6, 7, 8], [8, 7, 6, 5, 4, 3, 2, 1], [5, 4, 3, 2, 1])
      Exit: reverse([5, 6, 7, 8], [8, 7, 6, 5, 4, 3, 2, 1], [4, 3, 2, 1])
      Exit: reverse([4, 5, 6, 7, 8], [8, 7, 6, 5, 4, 3, 2, 1], [3, 2, 1])
      Exit: reverse([3, 4, 5, 6, 7, 8], [8, 7, 6, 5, 4, 3, 2, 1], [2, 1])
      Exit: reverse([3, 4, 5, 6, 7, 8], [8, 7, 6, 5, 4, 3, 2, 1], [2, 1])
      Exit: reverse([2, 3, 4, 5, 6, 7, 8], [8, 7, 6, 5, 4, 3, 2, 1], [1])
      Exit: reverse([1, 2, 3, 4, 5, 6, 7, 8], [8, 7, 6, 5, 4, 3, 2, 1], [])
Prev = [8, 7, 6, 5, 4, 3, 2, 1]
```

OUTPUT:

```
Prev = [8, 7, 6, 5, 4, 3, 2, 1]
```

DRYRUN:

The list [1,2,3,4,5,6,7,8] is load into memory than each item is copying in the second list sequentially like 1 is removed from first list and copied in the second from rear end in list then, 2 is removed from first list and copied in the second list but 2 is inserted at front end of the list then this procedure is repeated until the list1 became empty but the order in which the list 2 is coming is in descending order so, so that's how we get a reverse list. By using 2 lists.

QUESTION#15:

Maximum Number Code

```
1 max([A],A) :- !, true.
2 max([A|B], M):- max(B, M), M >= A.
3 max([A|B], A):- max(B, M), A >  M.
```

TRACE MODE IN QUERY WINDOW:

```
trace, (max([1,2,3,4],A)).
        Call: max([1, 2, 3, 4], _3974)
         Call: max([2, 3, 4], _3974)
          Call: max([3, 4], _3974)
           Call: max([4], _3974)
            Call: true
            Exit: true
           Exit: max([4], 4)
           Call: 4>=3
           Exit: 4>=3
          Exit: max([3, 4], 4)
          Call: 4>=2
          Exit: 4>=2
         Exit: max([2, 3, 4], 4)
         Call: 4>=1
         Exit: 4>=1
        Exit: max([1, 2, 3, 4], 4)
```

OUTPUT:

```
A = 4
```

DRYRUN:

In this there is a condition which is comparing each number by putting each number in a separate list to find the max number in the first time condition become false so it put 1 into a new list then it compares all like this and when it got 4 for comparison it compare 4 with 1,2 and 3 and declare 4 as max

QUESTION#16:

Swap Numbers Code:

```
1 swap([],[]).
2 swap([X,Y|T],[Y,X|T]).
3 swap([Z|T1],[Z|T2]):-swap(T1,T2).
```

TRACE MODE IN QUERY WINDOW:

```
trace, (swap([7,6,8,9,10],Z)).
   Call: swap([7, 6, 8, 9, 10], _3694)
   Exit: swap([7, 6, 8, 9, 10], [6, 7, 8, 9, 10])
```

OUTPUT:

```
Z = [6, 7, 8, 9, 10]
```

DRYRUN:

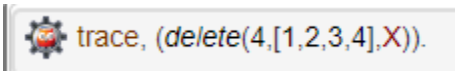
Because we have given the flip condition so this will swap the integers.

QUESTION#17:

Delete an element from list Code:

```
1 delete(A,[],[]).
2 delete(A,[A|As],As).
3 delete(A,[B|As],[B|Bs]) :- \+ A=B, delete(A,As,Bs).
```

TRACE MODE IN QUERY WINDOW:



```
  Call: delete(4, [1, 2, 3, 4], _3778)
   Call: 4=1
   Fail: 4=1
 Redo: delete(4, [1, 2, 3, 4], [1|_4090])
   Call: delete(4, [2, 3, 4], _4090)
    Call: 4=2
    Fail: 4=2
  Redo: delete(4, [2, 3, 4], [2|_4096])
    Call: delete(4, [3, 4], _4096)
     Call: 4=3
     Fail: 4=3
   Redo: delete(4, [3, 4], [3|_4102])
     Call: delete(4, [4], _4102)
     Exit: delete(4, [4], [])
    Exit: delete(4, [3, 4], [3])
   Exit: delete(4, [2, 3, 4], [2, 3])
  Exit: delete(4, [1, 2, 3, 4], [1, 2, 3])
```

OUTPUT:

```
X = [1, 2, 3]
```

QUESTION#18:

To find the list is sort or not Code:

```
1 sorted([]).
2 sorted([A]).
3 sorted([A,B|Bs]) :- A@<B, sorted([B|Bs]).
```

TRACE MODE IN QUERY WINDOW:

```
sorted([1,2,3,4]).
```

OUTPUT:

```
true
```

QUESTION#19:

To shift number >6 Code:

```
1 shift([H|T],W_List,Max):-max([H|T],Max),Max>6,concatenate(T,[H],W_List).
2 concatenate([],L,L).
3 concatenate([H1|T1],L2,[H1|T3]):-concatenate(T1,L2,T3).
4 max([A],A):-!,true.
5 max([A|B],M):-max(B,M),M>=A.
6 max([A|B],A):-max(B,M),A>M.
```

TRACE MODE IN QUERY WINDOW

```
trace, (shift([1,6,7],X,N)).
```

```
   Call: shift([1, 6, 7], _5278, _5274)
    Call: max([1, 6, 7], _5274)
     Call: max([6, 7], _5274)
      Call: max([7], _5274)
       Call: true
       Exit: true
      Exit: max([7], 7)
      Call: 7>=6
      Exit: 7>=6
     Exit: max([6, 7], 7)
     Call: 7>=1
     Exit: 7>=1
    Exit: max([1, 6, 7], 7)
    Call: 7>6
    Exit: 7>6
    Call: concatenate([6, 7], [1], _5278)
     Call: concatenate([7], [1], _5614)
      Call: concatenate([], [1], _5620)
```

```
   Call: concatenate([], [1], _5620)
   Exit: concatenate([], [1], [1])
  Exit: concatenate([7], [1], [7, 1])
 Exit: concatenate([6, 7], [1], [6, 7, 1])
Exit: shift([1, 6, 7], [6, 7, 1], 7)
```

OUTPUT:

N = 7,
X = [6, 7, 1]

QUESTION # 20:

A mammal is a class, a person shares the properties of the mammal, "Ram" is the instance of person who is the team member of team named XYZ having uniform of color blue. Person has a property that it has nose.

CODE:

```
1 person(mammal).
2 nose(person).
3 ram(person).
4 member(ram,team).
5 team(xyz,blueuniform).
```

OUTPUT:

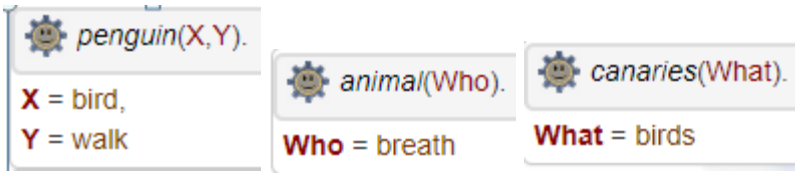
QUESTION # 21:

b) A bird has wings and can fly, since birds have wings and can fly, and canaries is also a type of birds, it seems reasonable that they also have wings and can fly. A penguin is also a bird and they can travel through walk. A bird is animal, Tweedy is also an animal, and Animals breathes in Air.

CODE:

```
1 canaries(birds).
2 animal(breath).
3 birds(animal).
4 birds(wings,fly).
5 penguin(bird,walk).
6 tweedy(animal).
```

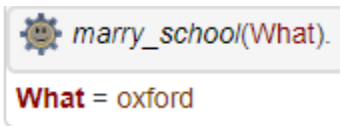
OUTPUT:



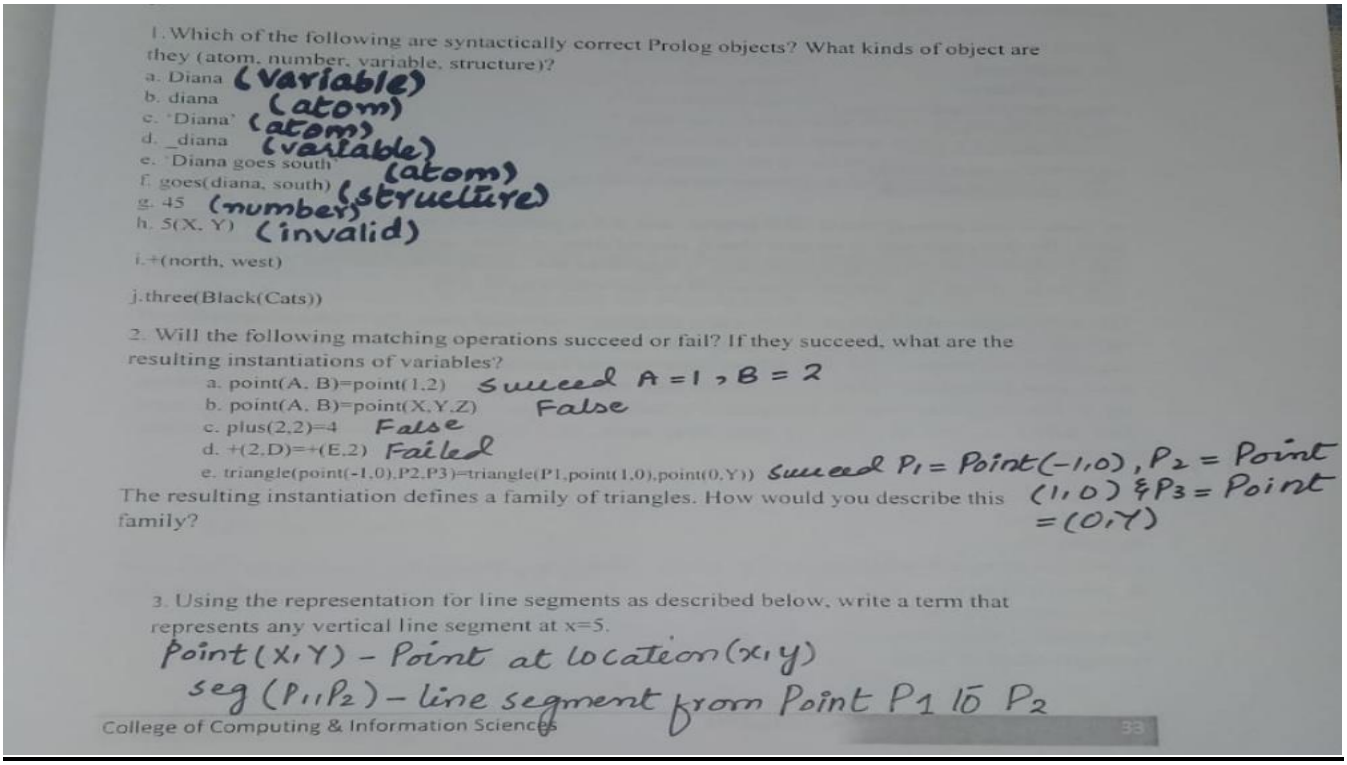
QUESTION # 22:

Robert is a former AI Scientist who works as a professor in Harvard University. He contributes in many projects like Web Development, AI, ML, Cloud Computing and Databases; He has a family with a wife named Ana who works as a Receptionist in a hospital, His 2 children named are John and Marry who study in a school which is near their home. The School named is Oxford.

CODE:



QUESTION #23:



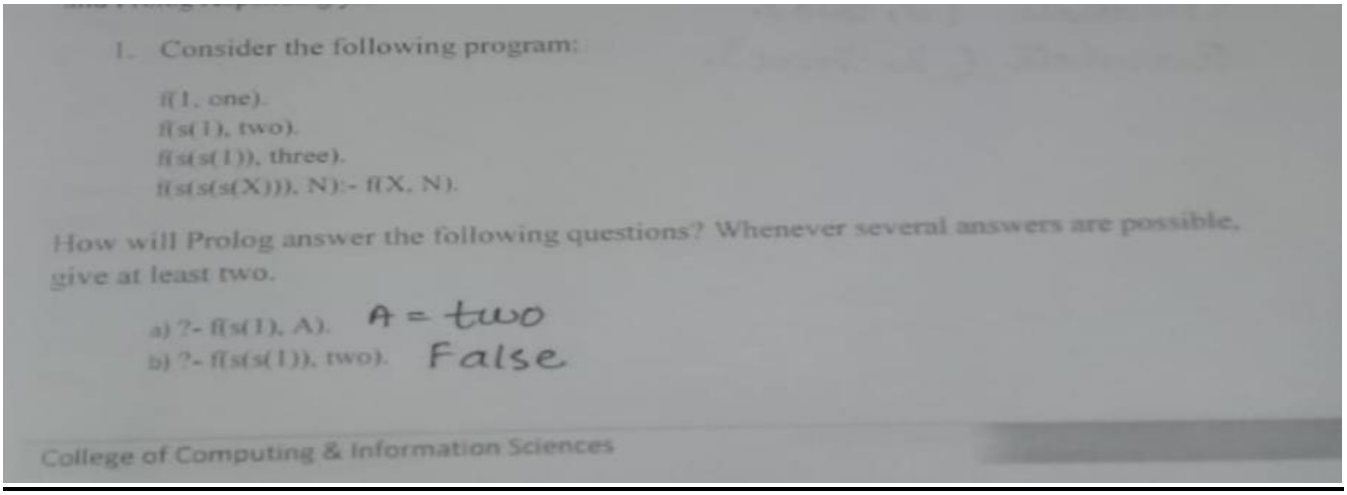
1. Which of the following are syntactically correct Prolog objects? What kinds of object are they (atom, number, variable, structure)?
a. Diana (variable)
b. diana (atom)
c. 'Diana' (atom)
d. _diana (variable)
e. 'Diana goes south' (atom)
f. goes(diana, south) (structure)
g. 45 (number)
h. 5(X, Y) (invalid)
i. +(north, west)
j. three(Black(Cats))

2. Will the following matching operations succeed or fail? If they succeed, what are the resulting instantiations of variables?
a. point(A, B)=point(1,2) Succeed A = 1, B = 2
b. point(A, B)=point(X,Y,Z) False
c. plus(2,2)=4 False
d. +(2,D)=+(E,2) Failed
e. triangle(point(-1,0),P2,P3)=triangle(P1,point(1,0),point(0,Y)) Succeed P1 = Point(-1,0), P2 = Point (1,0) & P3 = Point = (0,Y)
The resulting instantiation defines a family of triangles. How would you describe this family?

3. Using the representation for line segments as described below, write a term that represents any vertical line segment at x=5.
Point(X,Y) - Point at location(x,y)
seg(P1,P2) - line segment from Point P1 to P2

College of Computing & Information Sciences

QUESTION #24:



1. Consider the following program:

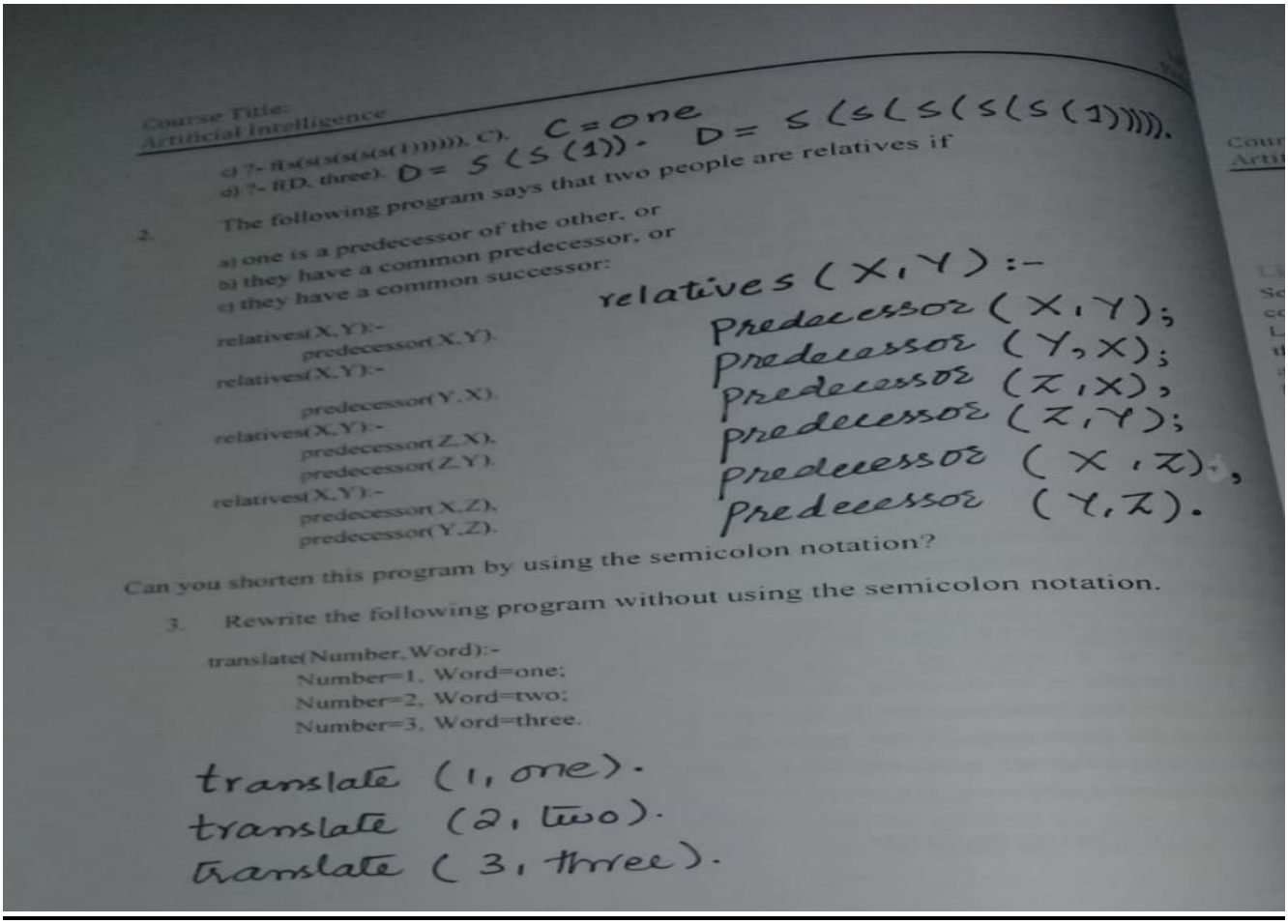
f(1, one).
f(s(1), two).
f(s(s(1)), three).
f(s(s(s(X))), N):- f(X, N).

How will Prolog answer the following questions? Whenever several answers are possible, give at least two.

a) ?- f(s(1), A). A = two
b) ?- f(s(s(1)), two). False

College of Computing & Information Sciences

QUESTION #24:



Course Title:
Artificial Intelligence

c) ?- f(s(s(s(s(s(s(1)))))), C). C = one
d) ?- f(D, three). D = s(s(1)). D = s(s(s(s(s(s(1)))))).

2. The following program says that two people are relatives if

a) one is a predecessor of the other, or
b) they have a common predecessor, or
c) they have a common successor:

```
relatives(X,Y):-
        predecessor(X,Y).
relatives(X,Y):-
        predecessor(Y,X).
relatives(X,Y):-
        predecessor(Z,X),
        predecessor(Z,Y).
relatives(X,Y):-
        predecessor(X,Z),
        predecessor(Y,Z).
```

relatives (X,Y) :-
Predecessor (X,Y);
Predecessor (Y,X);
Predecessor (Z,X),
Predecessor (Z,Y);
Predecessor (X,Z),
Predecessor (Y,Z).

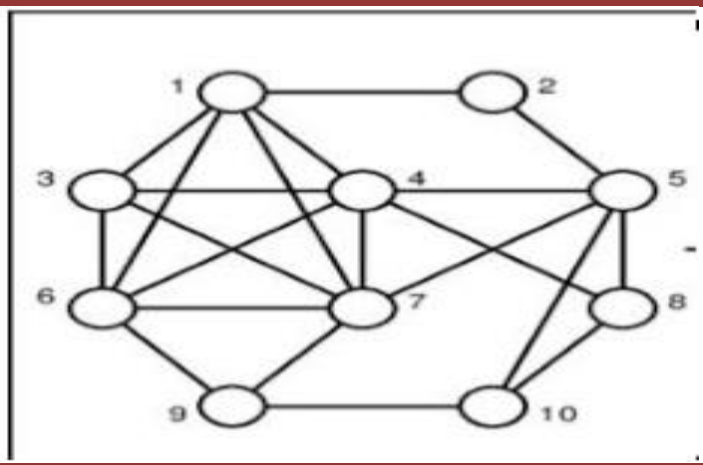
Can you shorten this program by using the semicolon notation?

3. Rewrite the following program without using the semicolon notation.

```
translate(Number,Word):-
        Number=1, Word=one;
        Number=2, Word=two;
        Number=3, Word=three.
```

translate (1, one).
translate (2, two).
translate (3, three).

QUESTION#25:



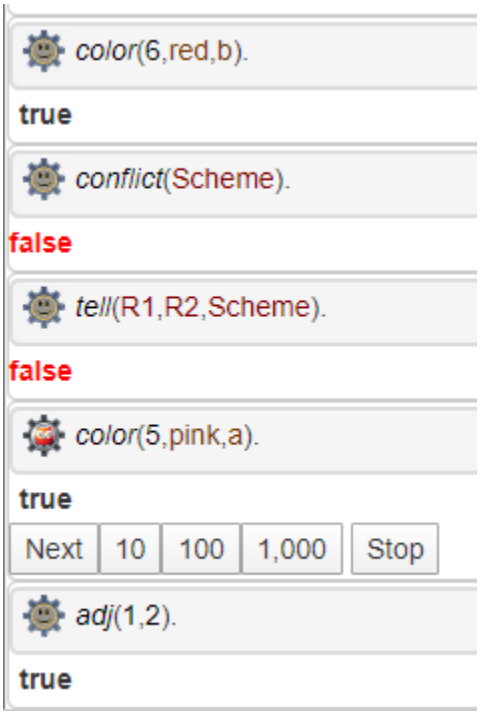
FIRST SOURCE CODE

```
adj(1,3).
adj(1,2).
adj(1,4).
adj(1,6).
adj(1,7).
adj(8,5).
adj(8,4).
adj(8,10).
adj(2,1).
adj(2,5).
adj(3,1).
adj(3,6).
adj(3,4).
adj(9,6).
```

```
adj(9,7).
adj(9,10).
adj(3,7).
adj(4,1).
adj(4,3).
adj(4,6).
adj(4,5).
adj(10,5).
adj(10,8).
adj(10,9).
adj(4,8).
adj(4,7).
adj(5,2).
adj(5,4).
adj(5,7).
adj(5,10).
adj(5,8).
adj(6,3).
adj(6,1).
adj(6,4).
adj(6,7).
adj(6,9).
adj(7,4).
adj(7,5).
adj(7,9).
adj(7,1).
adj(7,3).
adj(7,6).
color(1,pink,a).
color(5,pink,a).
color(9,pink,a).
color(7,yellow,a).
color(2,yellow,a).
color(10,yellow,a).
color(8,green,a).
color(3,green,a).
color(4,brown,a).
color(6,red,a).
```

```
54 color(1,pink,b).
55 color(5,pink,b).
56 color(9,pink,b).
57 color(7,yellow,b).
58 color(2,yellow,b).
59 color(10,yellow,b).
60 color(8,green,b).
61 color(3,green,b).
62 color(4,brown,b).
63 color(6,red,b).
64
65 conflict(Scheme):-adj(X,Y),color(X,Coloring,Scheme),color(Y,Coloring,Scheme).
66 tell(R1,R2,Scheme):-adj(R1,R2),color(R1,Coloring,Scheme),color(R2,Coloring,Scheme).
67
```

OUTPUT 1



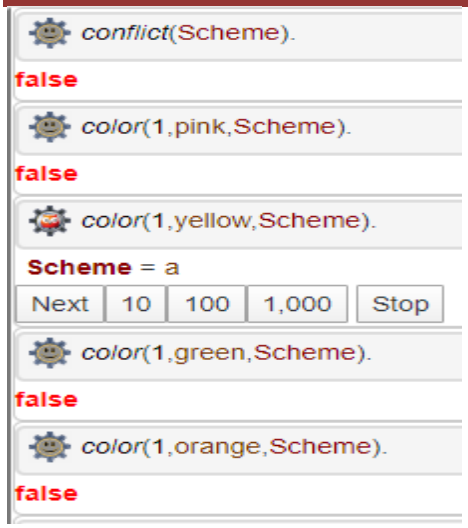
SECOND SOURCE CODE

```
adj(1,3).
adj(1,2).
adj(1,4).
adj(1,6).
adj(1,7).
adj(8,5).
adj(8,4).
adj(8,10).
adj(2,1).
adj(2,5).
adj(3,1).
adj(3,6).
adj(3,4).
adj(9,6).
adj(9,7).
adj(9,10).
adj(3,7).
adj(4,1).
adj(4,3).
adj(4,6).
adj(4,5).
adj(10,5).
adj(10,8).
adj(10,9).
adj(4,8).
adj(4,7).
```

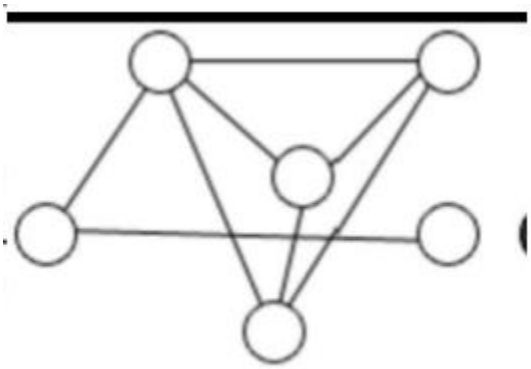
```
adj(5,2).
adj(5,4).
adj(5,7).
adj(5,10).
adj(5,8).
adj(6,3).
adj(6,1).
adj(6,4).
adj(6,7).
adj(6,9).
adj(7,4).
adj(7,5).
adj(7,9).
adj(7,1).
adj(7,3).
adj(7,6).
```

```
43 color(1,yellow,a).
44 color(5,yellow,a).
45 color(9,yellow,a).
46 color(7,pink,a).
47 color(2,pink,a).
48 color(10,pink,a).
49 color(8,brown,a).
50 color(3,brown,a).
51 color(4,green,a).
52 color(6,orange,a).
53
54 color(1,yellow,b).
55 color(5,yellow,b).
56 color(9,yellow,b).
57 color(7,pink,b).
58 color(2,pink,b).
59 color(10,pink,b).
60 color(8,brown,b).
61 color(3,brown,b).
62 color(4,green,b).
63 color(6,orange,b).
64
65 conflict(Scheme):-adj(X,Y),color(X,Coloring,Scheme),color(Y,Coloring,Scheme).
66 tell(R1,R2,Scheme):-adj(R1,R2),color(R1,Coloring,Scheme),color(R2,Coloring,Scheme).
```

OUTPUT 2



QUESTION#26:



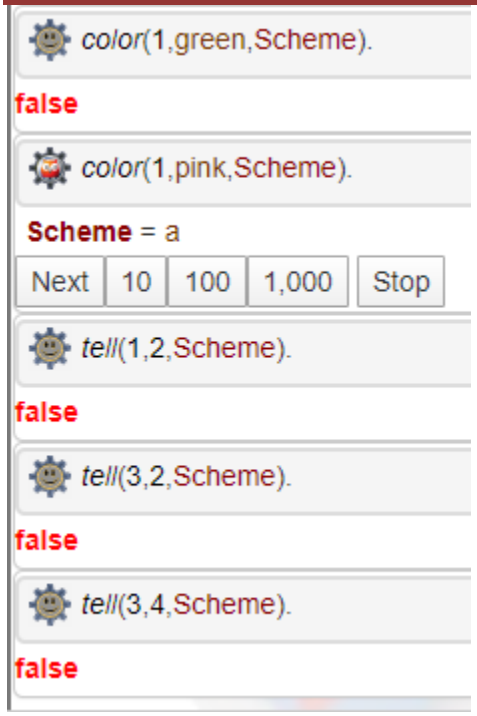
non-colored graph

FIRST SOURCE CODE

```
adj(1,2).
adj(1,3).
adj(1,4).
adj(1,5).
adj(2,1).
adj(2,3).
adj(2,5).
adj(3,1).
adj(3,2).
adj(3,5).
adj(4,1).
adj(4,6).
adj(5,3).
adj(5,2).
adj(5,1).
adj(6,4).
color(6,pink,a).
color(1,pink,a).
color(4,yellow,a).
color(2,yellow,a).
color(3,purple,a).
color(5,green,a).
color(6,pink,b).
color(1,pink,b).
color(4,yellow,b).
color(2,yellow,b).
```

```
27 color(3,purple,b).
28 color(5,green,b).
29 conflict(Scheme):-adj(X,Y),color(X,Coloring,Scheme),color(Y,Coloring,Scheme).
30 tell(R1,R2,Scheme):-adj(R1,R2),color(R1,Coloring,Scheme),color(R2,Coloring,Scheme).
```

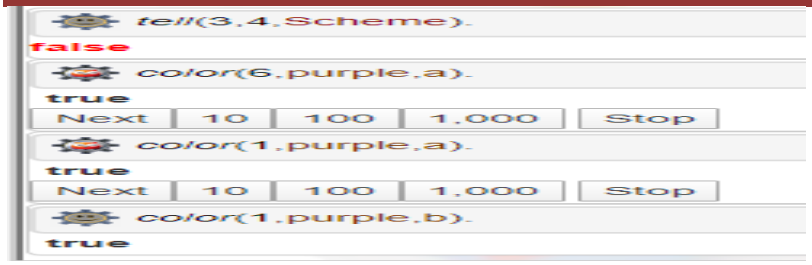
OUTPUT 3



SECOND SOURCE CODE

```
adj(1,2).
adj(1,3).
adj(1,4).
adj(1,5).
adj(2,1).
adj(2,3).
adj(2,5).
adj(3,1).
adj(3,2).
adj(3,5).
adj(4,1).
adj(4,6).
adj(5,3).
adj(5,2).
adj(5,1).
adj(6,4).
color(6,purple,a).
color(1,purple,a).
color(4,green,a).
color(2,green,a).
color(3,pink,a).
color(5,yellow,a).
color(6,purple,b).
color(1,purple,b).
color(4,green,b).
color(2,green,b).
color(3,pink,b).
color(5,yellow,b).
conflict(Scheme):-adj(X,Y),color(X,Coloring,Scheme),color(Y,Coloring,Scheme).
tell(R1,R2,Scheme):-adj(R1,R2),color(R1,Coloring,Scheme),color(R2,Coloring,Scheme).
```

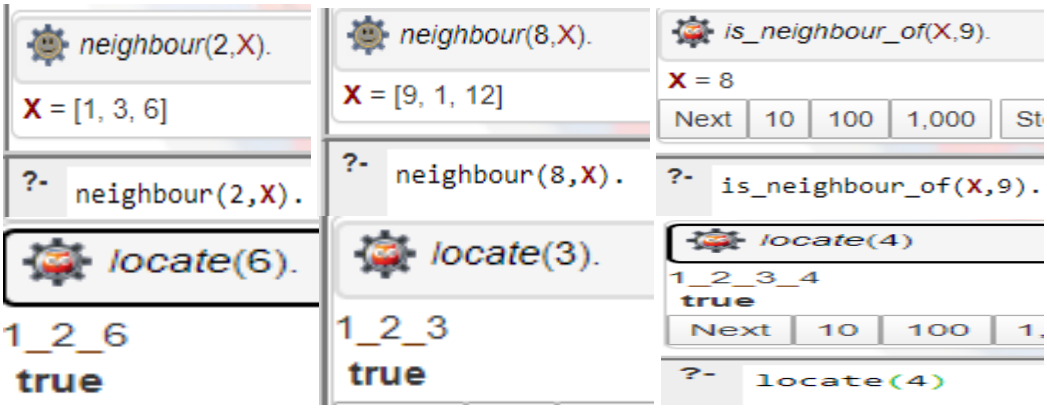
OUTPUT 4



QUESTION #27:

```
:-op(500,xfx,'is_neighbour_of').
1 is_neighbour_of 2.
1 is_neighbour_of 7.
1 is_neighbour_of 8.
2 is_neighbour_of 1.
2 is_neighbour_of 3.
4 is_neighbour_of 3.
5 is_neighbour_of 3.
2 is_neighbour_of 6.
7 is_neighbour_of 1.
8 is_neighbour_of 9.
8 is_neighbour_of 1.
8 is_neighbour_of 12.
3 is_neighbour_of 2.
3 is_neighbour_of 4.
3 is_neighbour_of 5.
6 is_neighbour_of 2.
9 is_neighbour_of 10.
9 is_neighbour_of 11.
9 is_neighbour_of 8.
12 is_neighbour_of 8.
11 is_neighbour_of 9.
10 is_neighbour_of 9.
neighbour(Node, Nodes) :- findall(Neighbour,Node is_neighbour_of Neighbour, Nodes).
path(1).
path(Node):-X is_neighbour_of Node,path(X),write(X),write('_').
locate(Node):-path(Node),write(Node),nl.
```

OUTPUT:



QUESTION #28(DFS IMPLEMENTATION):

DEPTH FIRST SEARCH

```
tree={
        '1':['2','7','8'],
        '2':['3','6'],
        '3':['4','5'],
        '4':[],
        '5':[],
        '6':[],
        '7':[],
        '8':['9','12'],
        '9':['10','11'],
        '10':[],
        '11':[],
        '12':[]
}
closed=set()
def dfs(closed,tree,node):
    if node not in closed:
        print(node)
        closed.add(node)
        for child in tree[node]:
            dfs(closed, tree, child)
dfs(closed,tree,'1')
```

OUTPUT:

```
1
2
3
4
5
6
7
8
9
10
11
12
```

QUESTION #29(BFS IMPLEMENTATION):

Breadth First Search

```
tree={
          1:[2,7,8],
          2:[3,6],
          3:[4,5],
          4:[],
          5:[],
          6:[],
          7:[],
          8:[9,12],
          9:[10,11],
          10:[],
          11:[],
          12:[]
}
def bfs(tree, initial):
     visited = []
     queue = [initial]
     while queue:
          node = queue.pop(0)
          if node not in visited:
               visited.append(node)
               neighbours = tree[node]
               for neighbour in neighbours:
                    queue.append(neighbour)
     return visited
print(bfs(tree,1))
```

OUTPUT:

```
[1, 2, 7, 8, 3, 6, 9, 12, 4, 5, 10, 11]
```

QUESTION #03

Shortest Path through FINDER:

```
tree={
        1:[2,7,8],
        2:[3,6],
        3:[4,5],
        4:[],
        5:[],
        6:[],
        7:[],
        8:[9,12],
        9:[10,11],
        10:[],
        11:[],
        12:[]
}
def bfs_shortest_path(tree, start, goal):
    explored = []
    queue = [[start]]
    if start == goal:
        return "found Goal with shortest path, success"
    while queue:
        path = queue.pop(0)
        node = path[-1]
        if node not in explored:
            neighbours = tree[node]
            for neighbour in neighbours:
                new_path = list(path)
                    new_path.append(neighbour)
                    queue.append(new_path)
                    if neighbour == goal:
                        return new_path
            explored.append(node)
    return "path not exist"
bfs_shortest_path(tree, 1, 12)
```

OUTPUT:

```
Out[23]: [1, 8, 12]
```

```
bfs_shortest_path(tree, 1, 6)
```

```
Out[24]: [1, 2, 6]
```

QUESTION #04

BAG OF WORDS IMPLEMENTATION

```
import numpy
import re
from nltk.corpus import stopwords
set(stopwords.words('english'))
def word_extract(sentence):
    ignore = ['a', "the", "is"]
    words = re.sub("[^\w]", " ",  sentence).split()
    cleaned_text = [w.lower() for w in words if w not in ignore]
    return cleaned_text

def tokenize(sentences):
    words = []
    for sentence in sentences:
        w = word_extract(sentence)
        words.extend(w)
        words = sorted(list(set(words)))
        return words

def BOW(allsentences):
    vocab = tokenize(allsentences)
    print("Word List for Document \n{0} \n".format(vocab))
    for sentence in allsentences:
        words = word_extract(sentence)
        bag_vector = numpy.zeros(len(vocab))
        for w in words:
            for i,word in enumerate(vocab):
              if word == w:
                  bag_vector[i] += 1
                  print("{0}\n{1}\n".format(sentence,numpy.array(bag_vector)))

file = ['I am Jaweria Asif. My hobby is writting stories',
       'I like to study',
       'I have my own visions to implement']
BOW(file)
```

OUTPUT:

```
Word List for Document
['am', 'asif', 'hobby', 'i', 'jaweria', 'my', 'stories', 'writting']

I am Jaweria Asif. My hobby is writting stories
[0. 0. 0. 1. 0. 0. 0. 0.]

I am Jaweria Asif. My hobby is writting stories
[1. 0. 0. 1. 0. 0. 0. 0.]

I am Jaweria Asif. My hobby is writting stories
[1. 0. 0. 1. 1. 0. 0. 0.]

I am Jaweria Asif. My hobby is writting stories
[1. 1. 0. 1. 1. 0. 0. 0.]

I am Jaweria Asif. My hobby is writting stories
[1. 1. 0. 1. 1. 1. 0. 0.]

I am Jaweria Asif. My hobby is writting stories
[1. 1. 1. 1. 1. 1. 0. 0.]

I am Jaweria Asif. My hobby is writting stories
[1. 1. 1. 1. 1. 1. 0. 1.]

I am Jaweria Asif. My hobby is writting stories
[1. 1. 1. 1. 1. 1. 1. 1.]

I like to study
[0. 0. 0. 1. 0. 0. 0. 0.]

I have my own visions to implement
[0. 0. 0. 1. 0. 0. 0. 0.]

I have my own visions to implement
[0. 0. 0. 1. 0. 1. 0. 0.]
```

QUESTION #05

IMPLEMENTATION OF COSINE SIMILARITY

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
X ="We are enrolled in BSCS, it is an amazing field"
Y ="Although BSCS is tough but it is interesting"
X_list = word_tokenize(X)
Y_list = word_tokenize(Y)
sw = stopwords.words('english')
l1 =[];l2 =[]
X_set = {w for w in X_list if not w in sw}
Y_set = {w for w in Y_list if not w in sw}
rvector = X_set.union(Y_set)
for w in rvector:
    if w in X_set: l1.append(1)
    else: l1.append(0)
    if w in Y_set: l2.append(1)
    else: l2.append(0)
c = 0
for i in range(len(rvector)):
        c+= l1[i]*l2[i]
cosine = c / float((sum(l1)*sum(l2))**0.5)
print("similarity: ", cosine)
```

OUTPUT:

```
similarity:  0.20412414523193154
```