Artificial Intelligence

SEM-7 No. 142

Lab-1

AIM: Introduction to Prolog and Study of Facts, Objects, Predicates and Variables.

ASSIGNMENT:

EX-1: Write a prolog program for the following facts.

i. Colour of b1 is red
ii. Colour of b2 is blue
iii. Colour of b3 is yellow
iv. Shape of b1 is square
v. Shape of b2 is circle
vi. Shape of b3 is square
vii. Size of b1 is small
viii. Size of b2 is small
ix. Size of b3 is large

What will be the outcome of each of the following queries?

i. What is the shape of b3?
ii. Which component is having large size and yellow colour?

PROGRAM:

```
domains
        prop,name,value=symbol
predicates
        find(prop,name,value)
clauses
        find(colour,b1,red).
        find(colour,b2,blue).
        find(colour,b3,yellow).
```

```
find(shape,b1,square).
find(shape,b2,circle).
find(shape,b3,square).
find(size,b1,small).
find(size,b2,small).
find(size,b3,large).
```

QUESTIONS:

i. What is the shape of b3?

INPUT: find(shape,b3,X)

OUTPUT:

ANS: square

ii. Which component is having large size and yellow colour?

INPUT: find(size,X,large) and find(colour,X,yellow)

OUTPUT:

ANS: b3

EX-2: Here are some simple clauses.

```
likes(mary,food).
likes(mary,wine).
likes(john,wine).
likes(john,mary).
```

The following queries yield the specified answers.

```
| ?- likes(mary,food).
yes.
| ?- likes(john,wine).
yes.
```

| ?- likes(john,food).
no.

How can you answer following questions?

1. John likes anything that Mary likes
2. John likes anyone who likes wine

PROGRAM:

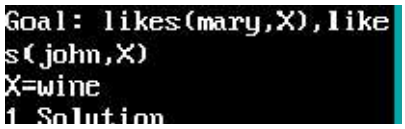
```
domains
        name1,name2=symbol
predicates
        likes(name1,name2)
clauses
        likes(mary,food).
        likes(mary,wine).
        likes(john,wine).
        likes(john,mary).
```

QUESTIONS:

1. John likes anything that Mary likes

INPUT: likes(mary,X),likes(john,X)

OUTPUT:

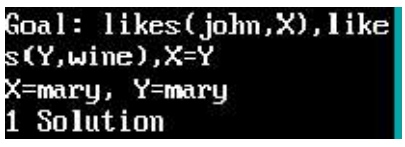


ANS: John likes wine as Mary likes wine.

2. John likes anyone who likes wine

INPUT: likes(john,X),likes(Y,wine),X=Y

OUTPUT:



ANS: John likes Mary because she likes wine.

EX-3: Here are some simple clauses.

has(jack,apples).
has(ann,plums).
has(dan,money).
fruit(apples).
fruit(plums).

How can you answer following questions?

1. What Jack has?
2. Does Jack have something?
3. Who has apples and Who has plums?
4. Does someone have apples and plums?
5. Has Dan fruits?

PROGRAM:

```
domains
        name1,name2=symbol
predicates
        has(name1,name2)
        fruit(name2)
clauses
        has(jack,apples).
        has(ann,plums).
        has(dan,money).
        fruit(apples).
        fruit(plums).
```

QUESTIONS:

1. What Jack has?

INPUT: has(jack,X)

OUTPUT:

```
Goal: has(jack,X)
X=apples
1 Solution
```

ANS: Jack has apples.

2. Does Jack have something?

INPUT: has(jack,_)

OUTPUT:

ANS: Yes, Jack has something(i.e apples).

3. Who has apples and Who has plums?

INPUT: has(X,apples),has(Y,plums)

OUTPUT:

ANS: Jack has apples and Ann has plums.

4. Does someone have apples and plums?

INPUT: has(X,apples) and has(X,plums)

OUTPUT:

ANS: No, no one have apples and plums.

5. Has Dan fruits?

INPUT: has(dan,X),fruit(X)

OUTPUT:

ANS: No, Dan does not have fruits.

LAB-2

AIM: Study of RULES & UNIFICATION.

ASSIGNMENT:

EX-1: Write a prolog program for the following facts and rules and answer the given question.

Facts:

i.	Parva has symptom fever
ii.	Parva has symptom rash
iii.	Parva has symptom headache
iv.	Parva has symptom runny nose
v.	Vidhi has symptom chills
vi.	Vidhi has symptom fever
vii.	Vidhi has symptom headache
viii.	Vivan has symptom runny nose
ix.	Vivan has symptom rash
x.	Vivan has symptom flu

Rules:

1.	Patient has Disease measles if Patient has symptoms fever, cough, conjunctivitis and rash.
2.	Patient has Disease german measles if Patient has symptoms fever, headache, runny nose and rash.
3.	Patient has Disease flu if Patient has symptoms fever, headache, body-ache and chills.
4.	Patient has Disease common cold if Patient has symptoms headache, sneezing, sore throat, chills and runny nose.
5.	Patient has Disease mumps if Patient has symptoms fever and swollen glands.
6.	Patient has Disease chicken pox if Patient has symptoms fever, rash, body-ache and chills.

Program:

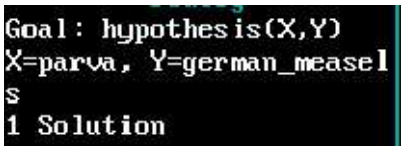
```
domains
        disease,indication,name=symbol
predicates
```

```
        hypothesis(name,disease)
        symptom(name,indication)
clauses

        symptom(parva,fever).
        symptom(parva,rash).
        symptom(parva,headache).
        symptom(parva,runny_nose).
        symptom(vidhi,chills).
        symptom(vidhi,fever).
        symptom(vidhi,headache).
        symptom(vivan,runny_nose).
        symptom(vivan,rash).
        symptom(vivan,flue).

        hypothesis(Patient,measels):-
                symptom(Patient,fever),
                symptom(Patient,cough),
                symptom(Patient,conjunctivitis),
                symptom(Patient,rash).

        hypothesis(Patient,german_measels):-
                symptom(Patient,fever),
                symptom(Patient,headache),
                symptom(Patient,runny_nose),
                symptom(Patient,rash).

        hypohesis(Patient,flu):-
                symptom(Patient,fever),
                symptom(Patient,headache),
                symptom(Patient,body_ache),
                symptom(Patient,chills).

        hypothesis(Patient,common_cold):-
                symptom(Patient,headache),
                symptom(Patient,sneezing),
                symptom(Patient,sore_throat),
                symptom(Patient,chills),
                symptom(Patient,runny_nose).

        hypothesis(Patient,mumps):-
```

```
        symptom(Patient,fever),
        symptom(Patient,swollen_glands).

    hypothesis(Patient,chicken_pox):-
        symptom(Patient,fever),
        symptom(Patient,rash),
        symptom(Patient,body_ache),
        symptom(Patient,chills).
```

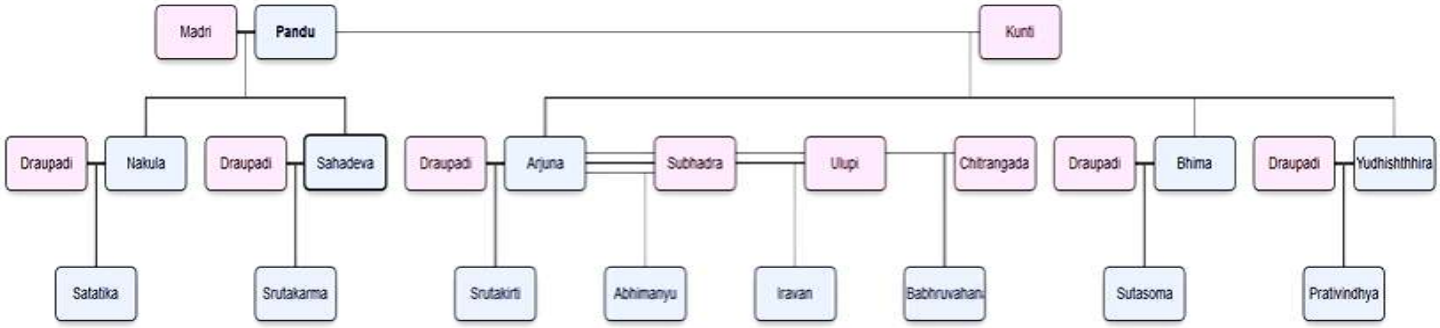
Questions:

1. Identify patient with any particular disease based on rules and facts given above.
Input: hypothesis(X,Y)
Output:



Answer: Parva has disease, “german measles”.

EX-2: Write a program for family tree given below which contains three predicates: male, female, parent. Make rules for family relations: father, mother, grandfather, grandmother, brother, sister, uncle, aunt, nephew and niece.



Program:

```
domains
        person=symbol
predicates
        male(person)
        female(person)
        parent(person,person)
        father(person,person)
```

```
        mother(person,person)
        grandfather(person,person)
        grandmother(person,person)
        brother(person,person)
        sister(person,person)
        uncle(person,person)
        aunt(person,person)
        nephew(person,person)
        niece(person,person)
clauses
        female(madri).
        female(draupadi).
        female(kunti).
        female(ulupi).
        female(subhadra).
        female(chitrangada).

         male(pandu).
        male(nakula).
        male(sahadeva).
        male(arjuna).
        male(bhima).
        male(yudhishthhira).
        male(satatika).
        male(srutakarma).
        male(sutasoma).
        male(prativindhya).
        male(abhimanyu).
        male(srutakirti).
        male(iravan).
        male(babhruvahana).

        parent(madri,nakula).
        parent(madri,sahadeva).
        parent(kunti,arjuna).
        parent(kunti,bhima).
        parent(kunti,yudhishthhira).
        parent(pandu,nakula).
        parent(pandu,sahadeva).
        parent(pandu,arjuna).
        parent(pandu,bhima).
```

```
parent(pandu,yudhishthhira).
parent(nakula,satatika).
parent(draupadi,satatika).
parent(sahadeva,srutakarma).
parent(draupadi,srutakarma).
parent(bhima,sutasoma).
parent(draupadi,sutasoma).
parent(yudhishthhira,prativindhya).
parent(draupadi,prativindhya).
parent(arjuna,srutakirti).
parent(arjuna,abhimanyu).
parent(arjuna,iravan).
parent(arjuna,babhruvahana).
parent(draupadi,srutakirti).
parent(subhadra,abhimanyu).
parent(ulupi,iravan).
parent(chitrangada,babhruvahana).

father(X,Y):-
        parent(X,Y),
        male(X).
mother(X,Y):-
        parent(X,Y),
        female(X).
grandfather(X,Y):-
        father(Z,Y),
        father(X,Z),
        male(X).
grandmother(X,Y):-
        father(Z,Y),
        mother(X,Z),
        female(X).
brother(X,Y) :-
        father(Z,Y),
        father(Z,X),
        male(X),
        X<>Y.
sister(X,Y):-
        father(Z,X),
         father(Z,Y),
        female(X),
```

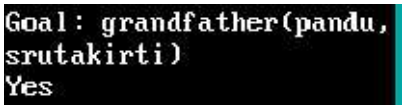
```
        X<>Y.
    uncle(X,Y):-
        parent(Z,Y),
        brother(X,Z),
        male(X).
    aunt(X,Y):-
        parent(Z,Y),
        sister(X,Z),
        female(X).
    nephew(X,Y):-
        uncle(Z,X),
        father(Z,Y),
        male(X).
    niece(X,Y):-
        uncle(Z,X),
        father(Z,Y),
        female(X).
```

Testing of Program:

1. **Father**

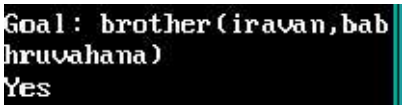
2. **Mother**

3. **Grand father**



4. **Grand mother**

5. **Brother**



6. **Sister**

```
Goal: sister(ulupi,subha
dra)
No
```

7. **Uncle**

```
Goal: uncle(bhima,prativ
indhya)
Yes
```

8. **Aunt**

```
Goal: aunt(draupadi,srut
akirti)
No
```

9. **Nephew**

```
Goal: nephew(X,arjuna)
X=satatika
X=srutakarma
X=sutasoma
X=prativindhya
4 Solutions
```

10. **Neice**

```
Goal: niece(abhimanyu,ul
upi)
No
```

EX-3: Write a prolog program for the following facts and rules, and trace the given goals:

Facts:

i.	hardware is easy course
ii.	Books for hardware are available
iii.	logic is not easy course
iv.	graphics is easy course
v.	graphics has 8 credits
vi.	graphics has lab component
vii.	Books for database are available
viii.	Mary takes compilers

Rules:

1.	X takes Y, if Y is easy course and books for Y are available
2.	X takes Y, if Y has 8 credits and Y has lab component

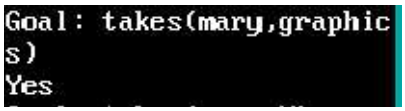
Program:

```
domains
        name,course,yn,feature=symbol
predicates
        person(name)
        available(course)
        easy(course,yn)
        has(course,feature)
        takes(name,course)
clauses
        person(mary).
        available(hardware).
        available(databases).
        easy(hardware,yes).
        easy(logic,no).
        easy(graphics,yes).
        has(graphics,eight_credit).
        has(graphics,lab_component).
        takes(X,Y):-
                easy(Y,yes),
                available(Y),
                person(X).
        takes(X,Y):-
                has(Y,eight_credit),
                has(Y,lab_component),
                person(X).
                takes(mary,compilers).
```

Goals:

a) Does Mary take graphics course?

Input: takes(mary,graphics)
Output:



Answer: Yes, Mary takes graphics course.

b) Which course Mary takes?

Input: takes(mary,X)
Output:

```
Goal: takes(mary,X)
X=hardware
X=graphics
X=compilers
3 Solutions
```

Answer: Mary takes hardware, graphics, and compilers courses.

c) Who takes graphics course?

Input: takes(X,graphics)
Output:

```
Goal: takes(X,graphics)
X=mary
1 Solution
```

Answer: Mary takes graphics course.

LAB-3

AIM: To learn simple input and output predicates in prolog and to build rule based consultation program.

ASSIGNMENT

EX-1: Build a rule based consulatation program for Medical Diagnosis

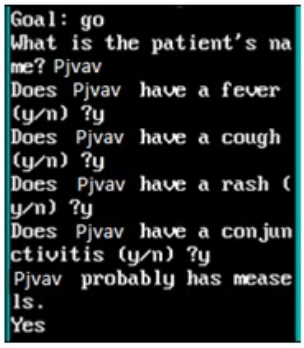
Program:

```
domains
        disease,indication = symbol
        Patient,name = string
predicates
        hypothesis(string,disease)
        symptom(name,indication)
        response(char)
        go
clauses
        go :-
                write("What is the patient's name? "),
                readln(Patient),
                hypothesis(Patient,Disease),
                write(Patient," probably has ",Disease,"."),nl.
```

```
go :-
        write("Sorry, I don't seem to be able to"),nl,
        write("diagnose the disease."),nl.
symptom(Patient,fever) :-
        write("Does ",Patient," have a fever (y/n) ?"),
        response(Reply),
        Reply='y'.
symptom(Patient,rash) :-
        write("Does ",Patient," have a rash (y/n) ?"),
        response(Reply),
        Reply='y'.
symptom(Patient,headache) :-

        write("Does ",Patient," have a headache (y/n) ?"),
        response(Reply),
        Reply='y'.
symptom(Patient,runny_nose) :-
        write("Does ",Patient," have a runny_nose (y/n) ?"),
        response(Reply),
        Reply='y'.
symptom(Patient,conjunctivitis) :-
        write("Does ",Patient," have a conjunctivitis (y/n) ?"),
        response(Reply),
        Reply='y'.
symptom(Patient,cough) :-
        write("Does ",Patient," have a cough (y/n) ?"),
        response(Reply),
        Reply='y'.
symptom(Patient,body_ache) :-
        write("Does ",Patient," have a body_ache (y/n) ?"),
        response(Reply),
        Reply='y'.
symptom(Patient,chills) :-
        write("Does ",Patient," have a chills (y/n) ?"),
        response(Reply),
        Reply='y'.
symptom(Patient,sore_throat) :-
        write("Does ",Patient," have a sore_throat (y/n) ?"),
        response(Reply),
        Reply='y'.
symptom(Patient,sneezing) :-
        write("Does ",Patient," have a sneezing (y/n) ?"),
```

```
        response(Reply),
        Reply='y'.

symptom(Patient,swollen_glands) :-
        write("Does ",Patient," have a swollen_glands (y/n) ?"),
        response(Reply),
        Reply='y'.
hypothesis(Patient,measels):-
        symptom(Patient,fever),
        symptom(Patient,cough),
        symptom(Patient,rash),
        symptom(Patient,conjunctivitis).
hypothesis(Patient,german_measels):-
        symptom(Patient,fever),
        symptom(Patient,rash),
        symptom(Patient,headache),
        symptom(Patient,runny_nose).
hypothesis(Patient,flu):-
        symptom(Patient,fever),
        symptom(Patient,body_ache),
        symptom(Patient,headache),
        symptom(Patient,chills).
hypothesis(Patient,common_cold):-
        symptom(Patient,sore_throat),
        symptom(Patient,sneezing),
        symptom(Patient,headache),
        symptom(Patient,chills),
        symptom(Patient,runny_nose).
hypothesis(Patient,chicken_pox):-
        symptom(Patient,fever),
        symptom(Patient,body_ache),
        symptom(Patient,rash),
        symptom(Patient,chills).
hypothesis(Patient,mumps):-
        symptom(Patient,fever),
        symptom(Patient,swollen_glands).
response(Reply) :-
        readchar(Reply), write(Reply),nl.
```

Input and Output:



EX-2: Predict user`s nature based on colour user likes.

Flow: Take user's name and asks for his/ her favourite colour using interactive questionnaire. Characteristic of colours are given below based on that predict user`s nature (i.e. aggressive, imaginative, cooperative, creative, introspective, affectionate, etc...)

No	Colour	Nature
1.	Red	It shows that you are very **social, assertive & energetic**. But at the same time, you are also **moody and impulsive**. You feel deep sympathy for fellow human beings and are easily swayed. You are an **optimist**, but you are also a **complainer** and do not desist from voicing your complaints or discomforts.
2.	Orange	You are **good natured**, enjoy being with others and are swayed by outside opinions. You do good work, have **strong loyalties**, and are very **good at heart**.
3.	Yellow	You are very **imaginative** and have a strong urge to help the world. You are inclined to speak of lofty ideas without applying them in practical. Secretly, you are **shy**, wish to be respected, crave admiration for your wisdom and are a mental loner. You are a **safe friend** in whom people can confide their secrets and problems.
4.	Green	You are a **good citizen** and a pillar of the community and are sensitive to social customs and etiquette. You are **frank, moral and reputable**. You make yourself a splendid teacher and feel deep affection for your family.

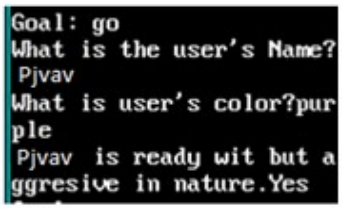
5.	Blue	You are **deliberate and introspective**. You have conservative convictions and retreat to gentler surroundings in times of stress, but are **sensitive** to the feelings of others. You keep a tight rein on your passions and enthusiasms, are a **loyal friend** and lead a sober life. You nourish big dreams but do not act on them. Stupidity in others annoys you, as does superior intelligence.
6.	Purple	You have a good mind, a **ready wit** and an ability to observe things that go unnoticed by others. You get **angry easily**. You display fine-arts creativity and appreciate the subtle but recognize the magnificent.
7.	Brown	You perform your duties very well, are **clever with money matters, stubborn** in your habits and convictions. You are **dependable** and steady, dislike impulsiveness and can bargain very well.
8.	Grey	You are **cautious**, try to strike a compromise in most situations. You encounter and seek composure and peace. You try very hard to fit yourself into a mould of your own design.
9.	Black	You are **above average, worldly, conventional, proper, polite and regal**. Black is a colour that means one thing (depression) to the clinical psychiatrist and quite another (dignity) to you.

Program:

```
domains
        Col,Name=string
        emotion=symbol
        predicates
        color(Name,Col)
        go
clauses
        go:-
                write("What is the user's Name? "),
                readln(Name),
                write("What is user's color?"),
                readln(Col),
                color(Name,Col).
        color(Name,red):-
                write(Name," is social,assertive,energetic,moody,impulsive,optimist,complainer in
nature.").
        color(Name,orange):-
                write(Name," is good in nature and good at heart and loyal in nature.").
        color(Name,yellow):-
                write(Name," is imaginative,shy and a safe friend in nature.").
        color(Name,green):-
```

```
        write(Name," is a good citizen and moral,frank and reputable in nature.").
    color(Name,blue):-
        write(Name," is deliberate,introspective,sensitive and loyal friend in nature.").

    color(Name,purple):-
        write(Name," is ready wit but aggresive in nature.").
    color(Name,brown):-
        write(Name," is clever with money matters but stubborn and dependable in
nature.").
    color(Name,gray):-
        write(Name," is cautious in nature.").
    color(Name,black):-
        write(Name," is above average,wordly,conventional,proper,polite and regal in
nature.").
```

Input and Output:



EX-3: Predict user`s health based on
Flow: Take user`s name and asks ve

EX-3: Predict user`s health based on habits user practices.
Flow: Take user`s name and asks yes/no for regular habits. Based on habits user follows regularly predict user`s health.

Health is considered to be bad if

- ➢ User has habit of regular smoking.
- ➢ User has habit of excessive drinking regularly.
- ➢ User has habit of taking drugs.
- ➢ User has habit of eating oily food and taking too much sugar with foods.
- ➢ User acts like an owl (i.e. Sleep hours are quite less).

Health is considered to be good if

- ➢ User has habit of drinking milk regularly and User has habit of eating green vegetables and or eggs in meal and User has habit of drinking enough water during day.
- ➢ User has habit of regular exercise and regular sufficient sleep hours and regular walk.
- ➢ User has habit of brushing teeth and washing hair and using showers regularly

Health is considered to be moderate if

- ➢ User has habit of eating oily food and having regular walk.
- ➢ User has habit of food with excessive sugar and having regular walk.
- ➢ User has habit of eating oily food and doing regular exercise.
- ➢ User has habit of eating food with excessive sugar and taking walk.

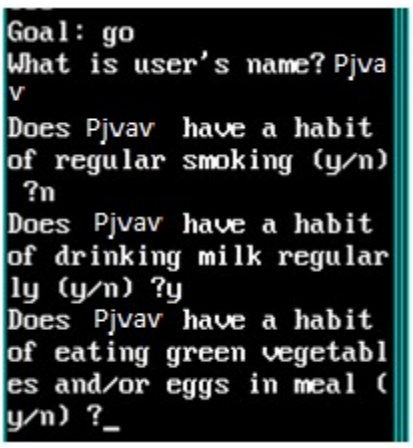
Program:

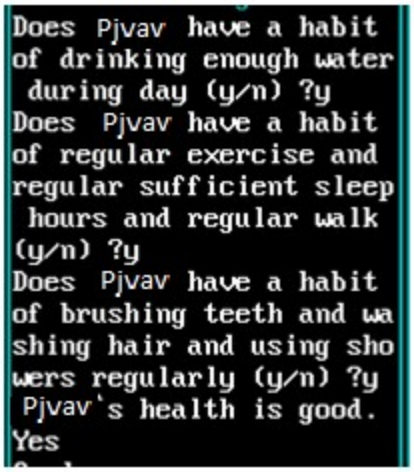
```
domains
        status,habit = symbol
        User,name = string
predicates
        health(string,status)
        have_habit(name,habit)
        response(char)
        go

clauses
        go :-
                write("What is user's name? "),
                readln(User),
                health(User,Status),
                write(User,"`s health is ",Status,"."),nl.
        go :-
                write("Sorry, I don't seem to be able to"),nl,
                write("predict user`s health status."),nl.
        have_habit(User,regular_smoking) :-
                write("Does ",User," have a habit of regular smoking (y/n) ?"),
                response(Reply),
                Reply='y'.
        have_habit(User,regular_excessive_drinking) :-
                write("Does ",User," have a regular habit of excessive drinking (y/n) ?"),
                response(Reply),
                Reply='y'.
        have_habit(User,drugs) :-
                write("Does ",User," have a habit of taking drugs (y/n) ?"),
                response(Reply),
                Reply='y'.
        have_habit(User,oily_sugar_food) :-
                write("Does ",User," have a habit of eating oily food and taking too much sugar
with foods(y/n) ?"),
```

```
                response(Reply),
                Reply='y'.
        have_habit(User,less_sleep) :-
                write("Does ",User," act like an owl i.e he/she sleeps quite less ?"),
                response(Reply),
                Reply='y'.

        have_habit(User,regular_milk) :-
                write("Does ",User," have a habit of drinking milk regularly (y/n) ?"),
                response(Reply),
                Reply='y'.
        have_habit(User,eggs_vegetables) :-
                write("Does ",User," have a habit of eating green vegetables and/or eggs in meal
(y/n) ?"),
                response(Reply),
                Reply='y'.
        have_habit(User,enough_water) :-
                write("Does ",User," have a habit of drinking enough water during day (y/n) ?"),
                response(Reply),
                Reply='y'.
        have_habit(User,regular_exercise_sleep_walk) :-
                write("Does ",User," have a habit of regular exercise and regular sufficient sleep
hours andregular walk (y/n) ?"),
                response(Reply),
                Reply='y'.
        have_habit(User,body_care) :-
                write("Does ",User," have a habit of brushing teeth and washing hair and using
showersregularly (y/n) ?"),
                response(Reply),
                Reply='y'.
        have_habit(User,oilyfood_regularwalk) :-
                write("Does ",User," have a habit of eating oily food and having regular walk (y/n)
?"),
                response(Reply),
                Reply='y'.
        have_habit(User,sugarfood_regularwalk) :-
                write("Does ",User," have a habit of food with excessive sugar and having regular
walk (y/n)?"),
                response(Reply),
                Reply='y'.
        have_habit(User,oilyfood_regularexercise) :-
```

```
        write("Does ",User," have a habit of eating oily food and doing regular exercise
(y/n) ?"),
        response(Reply),
        Reply='y'.
    have_habit(User,sugarfood_walk) :-
        write("Does ",User," have a habit of food with excessive sugar and taking walk
(y/n) ?"),
        response(Reply),
        Reply='y'.
    health(User,bad):-
        have_habit(User,regular_smoking),
        have_habit(User,regular_excessive_drinking),
        have_habit(User,drugs),
        have_habit(User,oily_sugar_food),
        have_habit(User,less_sleep).
    health(User,good):-
        have_habit(User,regular_milk),
        have_habit(User,eggs_vegetables),
        have_habit(User,enough_water),
        have_habit(User,regular_exercise_sleep_walk),
        have_habit(User,body_care).
    health(User,moderate):-
        have_habit(User,oilyfood_regularwalk),
        have_habit(User,sugarfood_regularwalk),
        have_habit(User,oilyfood_regularexercise),
        have_habit(User,sugarfood_walk).
    response(Reply) :-
        readchar(Reply), write(Reply),nl.
```

Input and Output:





Lab - 4

Aim: To learn arithmetic operations and recursion in Prolog.

Exercise : 1

Write a prolog program to find roots (real roots only) of quadratic equations.

```
predicates

printX

delta(real,real,real)

clauses

printX:-

write("Enter value of A,B and C:"),nl,

readReal(A),

readReal(B),

readReal(C),

Delta = (B*B)-(4*A*C),

delta(Delta,A,B).

delta(Delta,A,B):-
```

```
Delta>=0,
X = (-B+sqrt(Delta))/2*A,
Y = (-B-sqrt(Delta))/2*A,
write("Root1: "),write(X),nl,
write("Root2: "),write(Y),nl.
```

Output :

```
Goal: printX
Enter values of a, b and c:
1
7
12
Root1: -3
Root2: -4
Goal: _
```

Exercise : 2

Write a prolog program to implement a logon routine. This routine must asks username and password and verify with pair of username and password available (i.e. stored as clauses) as facts. On successful match system display “welcome message” and on an unsuccessful attempt user is allowed 3 times to reenter valid credentials. If user enters incorrect credential continuously 3 times then system exits with “unsuccessful attempt message”.

```
Predicates
user(string,string)
```

```
Login

login(integer)

check(string,string,integer)

Clauses

login:- login(0).

login(3):- write("Unsuccessful Attempt!"),nl.

login(X):-X<3,

XX=X+1,write("Enter your username & password: "),nl,

readln(Username),readln(Password),check(Username,Password,XX).

check(Username,Password,XX):-user(Username,Password),write("Welcome "),

write(Username),nl.

check(Username,Password,XX):-

write("Incorrect Details!"),nl,login(XX).

user(prakruti,pjvavdiya).

user(user1,user10702).
```

Output :

```
Goal: login

Enter your username & password:

prakruti

prakruti

Incorrect Details!

Enter your username & password:

user1

user10000
```

```
Incorrect Details!
Enter your username & password:
prakruti
pjvavdiya
Welcome prakruti


Yes
Goal: _
```

Exercise : 3

Write a prolog program to find factorial of a given number.

```
predicates
fact(integer,integer)
clauses
fact(0,1).
fact(X,Ans):-
X>0,
X1 = X-1,
fact(X1,R1),
Ans = X * R1.
```

Output :

```
Goal: fact(0,X)
X=1
1 Solution
Goal: fact(4,X)
```

```
X=24

1 Solution

Goal: fact(5,X)

X=120

1 Solution

Goal: _
```

Exercise : 4

Write a prolog program to find sum of first n number.

```
predicates

sum(integer,integer)

clauses

sum(X,Ans):-

X>=0,

Ans = X * (X+1) / 2.
```

Output :

```
Goal: sum(5,X)

X=15

1 Solution

Goal: sum(10,X)

X=55

1 Solution

Goal: sum(100,X)

X=5050

1 Solution
```

```
Goal: _
```

Exercise : 5

Write a prolog program to print nth term of Fibonacci series.

```
predicates

fibo(integer,integer)

clauses

fibo(1,1).

fibo(2,1).

fibo(N,Ans):-

N>0,

N1 = N-1,

fibo(N1,R1),

N2 = N-2,

fibo(N2,R2),

Ans= R1+R2.
```

Output :

```
Goal: fibo(1,X)

X=1

1 Solution

Goal: fibo(3,X)

X=2

1 Solution

Goal: fibo(10,X)

X=55
```

1 Solution

Goal: _

Exercise : 6

Write a prolog program to print Fibonacci series up-to nth term.

```
predicates
fibo(integer,integer)
go(integer)
clauses
go(0):-
write("").
go(X):-
X1=X-1,
go(X1),
fibo(X,R),
write(R),nl.
fibo(1,1).
fibo(2,1).
fibo(N,Ans):-
N>0,
N1 = N-1,
fibo(N1,R1),
N2 = N-2,
fibo(N2,R2),
Ans= R1+R2.
```

Output :

```
Goal: go(5)

1 1 2 3 5

Goal: go(10)

1 1 2 3 5 8 13 21 34 55

Goal: _
```

Lab-5

AIM: To study about controlling execution in prolog using cut and fail predicate

EXERCISE

1. Implement a prolog program to find minimum and maximum of two integers using cut and/or fail predicate. Program must have three arguments and it must handle all cases.

Code:

```
predicates

max(integer,integer,integer)

clauses

max(X,Y,Z):- X>=Y, Z=X, !.

max(X,Y,Z):- X<Y,Z=Y.
```

Output:

```
Goal: max(6,8,X)

X=8

1 solution
```

```
Goal: max(4,5,2)

No
```

2. Write a prolog program to verify that given year is leap year or not using cut and/ or fail predicate.

Note: A year is a leap year if it is divisible by 4, but century years are not leap years unless they are divisible by 400. So, the years 1700, 1800, and 1900 were not leap years, but the year 2000 was.

Code:

```
domains

z = integer

predicates

leap(integer)

clauses

leap(X):- X mod 4<>0,!,fail.

leap(X):- X mod 100<>0,!.

leap(X):- X mod 400=0.
```

Output:

```
Goal: leap(2004)

Yes

Goal: leap(2019)

Yes
```

3. write a prolog program to check whether a number is prime or not.

Code:

```
Predicates

Prime(integer)
```

```
Check(integer,integer)

Clauses

prime(0):- !,fail.

prime(1):- !,fail.

prime(2):- !.

prime(X):- check(X,2).

Check(X,Y):- Y>sqrt(X),!.

Check(X,Y):- Y<=sqrt(X),X mod Y=0,!,fail.

Check(X,Y):- YY=Y+1, check(X,YY).
```

Output:

```
Goal: prime(7)

Yes

Goal: prime(2)

Yes

Goal: prime(6)

No
```

Lab - 6

AIM: Study LIST structure in PROLOG

1. write a prolog program to check number is in list or not.

```
Domains

        list=integer*
```

```
Predicates
        findnum(integer,list)
Clauses
        findnum(X,[]):-
                write(\"\\nNumber Is Not Found\").
        findnum(X,[X|Tail]):-
                write(\"\\nNumber Is Found\").
        findnum(X,[Y|Tail]):-
                findnum(X,Tail).
```

OUTPUT:

```
Goal : findnum(3,[1,2,3])
Output : Number is found
Yes
Goal: findnum(4,[1,2,3])
Output:Number not found
No
```

2. Write a prolog program for concatenating list in third list

```
Domains
        list=integer*
Predicates
        concat_list(list,list,list)
Clauses
        concat_list([],L1,L1).
```

```
concat_list([X|Tail],L2,[X|Tail1]):-
        concat_list(Tail,L2,Tail1).
```

OUTPUT:

```
Goal:
concat_list([1,2],[3,4],ConcatList)
Output:
ConcatList=[1,2,3,4]
1 Solution
```

3. Write a prolog program for finding last element in list

```
Domains
        list=integer*
Predicates
        find_last(list)
Clauses
        find_last([X]):-
                write(\"\\nLast element is : \"),
                   write(X).
        find_last([Y|Tail]):-
                    find_last(Tail).
```

OUTPUT:

```
Goal :
find_last([1,2,3,4])
Output:
```

```
Last element is 4

Yes
```

4. Write a prolog program to reverse a list.

```
Domains
        list=integer*
Predicates
        rev_list(list)
        findrev(list,list,list)
Clauses
        rev_list(L):-
                findrev(L,[],List2),
                  write(\"\\nReverse Of Given List : \",List2).
                findrev([],List1,List1).
        findrev([X|Tail],List1,List2):-
                   findrev(Tail,[X|List1],List2).
```

OUTPUT:

```
Goal : rev_list([1,5,3,2]).

Output : Reverse of given list [2,3,5,1]).

Yes
```

5. Write a prolog program to find nth element of a given list.

```
domains
        num = integer
        list= integer*
predicates
```

```
        find_num(list,num)
Clauses
        find_num([],N) :-
                write("empty list"),nl.
        find_num([E|List],1) :-
                write(\"The element is \",E),nl.
        find_num([E|List],N) :-
                N1 = N-1,
                find_num(List,N1).
```

OUTPUT:

```
Goal:find_num([1,2,3,4],3).
Output: The Element is 3.
```

6. **write a prolog program to split list in two list such that one list contains negative numbers and one positive numbers.**

```
domains
        list=integer*
predicates
        split_list(list,list,list)
clauses
        split_list([],[],[]).
        split_list([X|L],[X|L1],L2):-
                X>= 0,
                !,
```

```
        split_list(L,L1,L2).

split_list([X|L],L1,[X|L2]):-

        split_list(L,L1,L2) .
```

OUTPUT:

Goal: spilt_list([3,6,-2,6,-33],X,Y).

OutPut: X=[3,6,6], Y=[-2,-33]

1 Solution

LAB - 7

AIM: Study Compound Object and Functors in PROLOG.

1. Modify the sample program II so that it will also print the birth dates of the people listed. Next, add telephone numbers to the report.

```
DOMAINS
        name = person(symbol,symbol)
        birthday = b_date(symbol,integer,integer)

PREDICATES
        phone_list(name,symbol,birthday)
        get_months_birthdays
        convert_month(symbol,integer)
        check_birthday_month(integer,birthday)
        write_person(name)
        write_phone(symbol)
        write_birthdate(birthday)
CLAUSES
        get_months_birthdays:-
```

```
        write("************ This Month's Birthday List ************"),nl,
        write(" First name Last Name\tPhoneNo.\t BirthDate\n"),
        write("**************************************************"),nl,
        date(_, This_month, _),
        phone_list(Person,Phone, Date),
        check_birthday_month(This_month, Date),
        write_person(Person),write_phone(Phone),write_birthdate(Date),
        fail.

get_months_birthdays:-
        write("\n\n Press any key to continue: "),nl,
        readchar(_).

write_person(person(First_name,Last_name)):-
        write(" ",First_name," ",Last_name).
write_phone(Phone):- write("\t",Phone).

write_birthdate(b_date(Month,Day,Year)):-
        write("\t",Day,"-",Month,"-",Year),nl.

check_birthday_month(Mon,b_date(Month,_,_)):-
        convert_month(Month,Month1), Mon = Month1.

phone_list(person(apurva, mehta), "767-8463", b_date(jan, 13, 1955)).
phone_list(person(apurva, shah), "438-8400", b_date(feb, 04, 1985)).
phone_list(person(apurva, parikh), "555-5653", b_date(mar, 22, 1935)).
phone_list(person(apurva, doshi), "767-2223", b_date(apr, 04, 1951)).
phone_list(person(apurva, joshi), "555-1212", b_date(may, 31, 1962)).
phone_list(person(apurva, baxi), "438-8400", b_date(jun, 13, 1980)).
phone_list(person(apurva, dave), "767-8463", b_date(jun, 22, 1986)).
phone_list(person(apurva, bhatt), "555-5653", b_date(jul, 22, 1981)).
phone_list(person(apurva, patel), "767-2223", b_date(aug, 13, 1981)).
phone_list(person(apurva, dangar), "438-8400", b_date(sep, 22, 1981)).
phone_list(person(apurva, pandya), "438-8400", b_date(oct, 31, 1952)).
phone_list(person(apurva, vaishnav), "555-1212", b_date(nov, 22, 1984)).
phone_list(person(apurva, gor), "767-2223", b_date(nov, 04, 1987)).
```

```
    phone_list(person(apurva, kanani), "438-8400", b_date(dec, 31, 1981)).

    convert_month(jan, 1).
    convert_month(feb, 2).
    convert_month(mar, 3).
    convert_month(apr, 4).
    convert_month(may, 5).
    convert_month(jun, 6).
    convert_month(jul, 7).
    convert_month(aug, 8).
    convert_month(sep, 9).
    convert_month(oct, 10).
    convert_month(nov, 11).
    convert_month(dec, 12).
```

OUTPUT

```
Goal: get_months_birthdays
Output:-
********** This Month's Birthday List **********
First name   Last Name   PhoneNo     BirthDate
**************************************************
apurva       dangar      438-8400    22-sep-1981
Yes
```

2. Write a prolog program for an IT company that store employee details like Name, Address, Department, Position, Salary. Use compound objects to properly formulate the representation of each employee details. Find out

I. employee(s) with salary higher than a threshold

II. employee(s) available in a particular department

III. employee(s) holding a particular position

```
DOMAINS
    name=person(symbol,symbol)
    address=place(symbol,symbol,symbol)
    department=dept(symbol,symbol)
    Salary=real
```

```
predicates
        get_emp_by_salary(real)
        get_emp_by_dept(symbol)
        get_emp_by_position(symbol)
        employee(name,address,department,real)
        write_name(name)
        write_dept(department)
        write_salary(real)
clauses
        get_emp_by_salary(Salary):-
                write("************ Employee List ************"),nl,
                write(" First name Last Name\tDepartment.\t Position\tSalary\n"),
                write("*************************************************"),nl,
                employee(Name,_,Department,S),
                S>=Salary,write_name(Name),write_dept(Department),
                write_salary(S),fail.

        get_emp_by_salary(_):-
                write("\n\n Press any key to continue: "),nl,
                readchar(_).

        get_emp_by_dept(D):-
                write("************ Employee List ************"),nl,
                write(" First name Last Name\tSalary\t Department\tPosition\n"),
                write("*************************************************"),nl,
                employee(Name,_,dept(DName,P),S),DName=D,write_name(Name),
                write_salary(S),write_dept(dept(DName,P)),fail.

        get_emp_by_dept(_):-
                write("\n\n Press any key to continue: "),nl,
                readchar(_).

        get_emp_by_position(Position):-
                write("************ Employee List ************"),nl,
                write(" First name Last Name\tSalary\tDepartemnt\tPosition\n"),
                write("*************************************************"),nl,
```

```
        employee(Name,_,dept(DName,P),S),P=Position,write_name(Name),
        write_salary(S),write_dept(dept(DName,P)),fail.

    get_emp_by_position(_):-
        write("\n\n Press any key to continue: "),nl,
        readchar(_).

    write_name(person(Firstname,Lastname)):- write(Firstname," ",Lastname).
    write_dept(dept(Name,Position)):- write("\t",Name,"\t", Position).
    write_salary(Salary):- write("\t",Salary),nl.

    employee(person(tina,soni),place(anand,anand,gujarat),dept(software,developer),8000
0).
    employee(person(hardik,shah),place( bhuj,bhuj,gujarat ), dept(software ,asstDeveloper)
,30000).
    employee(person(janvi,patel),place(surat,surat,gujarat),dept(support,sde2),1000000).
```

OUTPUT

```
Goal: get_emp_by_salary(80000)
Output: ********** Employee List **********
First name Last Name Department Position Salary
**************************************************
tina soni software developer 80000
Yes
Goal: get_emp_by_dept(support)
First name Last Name Department Position Salary
**************************************************
Janvi patel support sde2 1000000
```

3. Try the following link and verify whether the system is intelligent or not and justify your answer.

www.manifestation.com/neurotoys/eliza.php3

ELIZA has almost no intelligence. It uses string substitution and canned responses based on keywords. It has a common set of keywords used to give responses. It keeps repeating certain set of responses if it does not understand what we type.

LAB – 8

AIM: Database Handling in Prolog

1. Write a prolog program to create a game like "KBC".

```
domains
        q,a,b,c,d=string
        ans=char
database
        ques(q,a,b,c,d,ans)
predicates
        play_game
        print_ques(q,a,b,c,d)
        start_game
        ask_question
        check_ans(char,char)
        check(char)
clauses
        play_game:-
                consult("results.txt"),
                makewindow(1,7,7,"Kaun Bangega Crorepati",0,0,25,80),
                write("Welcome to KBC"),nl,
                write("start game?1/0"),nl,
                readint(X),X=1,start_game.
                play_game.
        start_game:-
                ask_question.
        ask_question:-
                retract(ques(Q,A,B,C,D,Ans)),
                print_ques(Q,A,B,C,D),
```

```
        readchar(X),
        check_ans(X,Ans),
        write("Want TO continue?y/n"),
        readchar(Y),
        check(Y).
ask_question:-
        nl,write("Khel Sampat hua"),nl,
        write("Good Bye!"),
        readchar(_).
check(Y):-Y='y',!,fail.
check(_):-
        write("Khel Sampat hua"),
        write("Good Bye!"),
        readchar(_).
print_ques(Q,A,B,C,D):-
        write("Yeh raha apka agla saval.."),nl,
        write(Q),nl,
        write("a)",A), nl,
        write("b)",B),nl,
        write("c) ",C),nl,
        write("d)",D),nl,
        write("Enter your ans").
check_ans(X,Ans):-
        X=Ans,write("Sahi Javab!"),nl,!.
check_ans(_,_):-
        write("Galat Javab!!"),nl.
```

OUTPUT:

```
ques("capital of india?","new Delhi","Allahabad","Agra","Udaipur",'a').
ques("where is gujarat","Germany","Paris","New York","India",'d').
```

2. Write a prolog program to create an application like marriage beuro.

```
%trace
domains
        file = input
        personName,gender=string
        age=integer
        attributList=string*
database
```

```
        personData(personName,gender,age,attributList,attributList)
predicates
        repeat
        showMenu
        selection(integer)
        getCandidateDetail
        getAttribute(attributList)
        genderValidation(gender)
        ageValidation(age)
        searchCandidate(gender,age,personName,attributList)
        probalePairs
        readline
        goal
        clearwindow,
        showMenu.
clauses
        repeat.
        repeat:-repeat.
        getCandidateDetail:-
                write("\nEnter candidate details...."),
                write("\nEnter name of person : "),
                readln(Name),
                write("\nEnter the gender of a person(male/female) : "),
                readln(Gender),
                genderValidation(Gender),
                write("\nEnter the age of a person : "),
                readint(Age),
                not(ageValidation(Age)),
                write("\nEnter the attributes..."),
                getAttribute(AttributList),
                write("\nEnter the preferences..."),
                getAttribute(PreferenceList),
                assert(personData(Name,Gender,Age,AttributList,PreferenceList)).
        getAttribute(Attr):-
                write("\nEnter the qualification : "),
                readln(Qual),
                write("\nEnter the height : "),
                readln(Height),
                write("\nEnter the weight : "),
                readln(Weight),
                write("\nEnter the color : "),
```

```
        readln(Color),
        Attr=[Qual,Height,Weight,Color].
%validation
genderValidation(Gender):-
        Gender=male;Gender=female.
ageValidation(Age):-
        Age<18,
        write("\nEnter valid age...").
        %search for male candidate
searchCandidate(Gender,Age,Name,PreferenceList):-
        Gender="female",
        personData(Name,"male",TAge,AttributList,_),
        TAge >= Age,
        AttributList=PreferenceList.
        %search for female candidate
searchCandidate(Gender,Age,Name,PreferenceList):-
        Gender="male",
        personData(Name,"female",TAge,AttributList,_),
        TAge <= Age,
        AttributList=PreferenceList.
%find probale pairs
probalePairs:-
        personData(Fname,"female",Fage,FattributList,FpreferenceList),
        personData(Mname,"male",Mage,MattributList,MpreferenceList),
        Fage <= Mage,
        FattributList=MpreferenceList,
        MattributList=FpreferenceList,
        write("\n",Mname,"-------->",Fname).
%main menu
showMenu:-
        repeat,
        write("\n**********Marriege burro**********"),
        write("\n1.Enter candidate detail into database"),
        write("\n2.Search for candidate"),
        write("\n3.Probeble pairs of matching parteners"),
        write("\n4.Show all candidate details"),
        write("\n0.Exit"),
        write("\n****************************************"),
        write("\nEnter your choice::"),
        readint(Choice),
        selection(Choice),
```

```
        Choice=0.
readline:-
        not(eof(input)),
        readln(LineL),
        write(LineL),nl,
        readline.
selection(0).
selection(1):-
        % getCandidateDetail,
        openread(input,"marriage.txt"),
        readdevice(input),
        readline.
selection(2):- write("\nEnter the details of candidate..."),
        write("\nEnter gender : "),
        readln(Gender),
        gendervalidation(Gender),
        write("\nEnter age : "),
        readint(Age),
        not(ageValidation(Age)),
        write("\nEnter the detail of Patners..."),
        getAttribute(AttributList),
        write("\nMatching Partners are..."),
        searchCandidate(Gender,Age,Name,AttributList),
        write("\n",Name),
        fail.
selection(3):-
        probalePairs, fail.
selection(4):-
        personData(Name,Gender,Age,AttributList,PreferenceList),
        write("\n-------------------------------------------------------------------------"),
        write("\nName : ",Name),
        write("\nGender : ",Gender),
        write("\nAge : ",Age),
        write("\nAttribute[Qualification,Height,Weight,Color] : ",AttributList),
        write("\nPreferece[Qualification,Height,Weight,Color] : ",PreferenceList),
        write("\n-------------------------------------------------------------------------").
```

OUTPUT

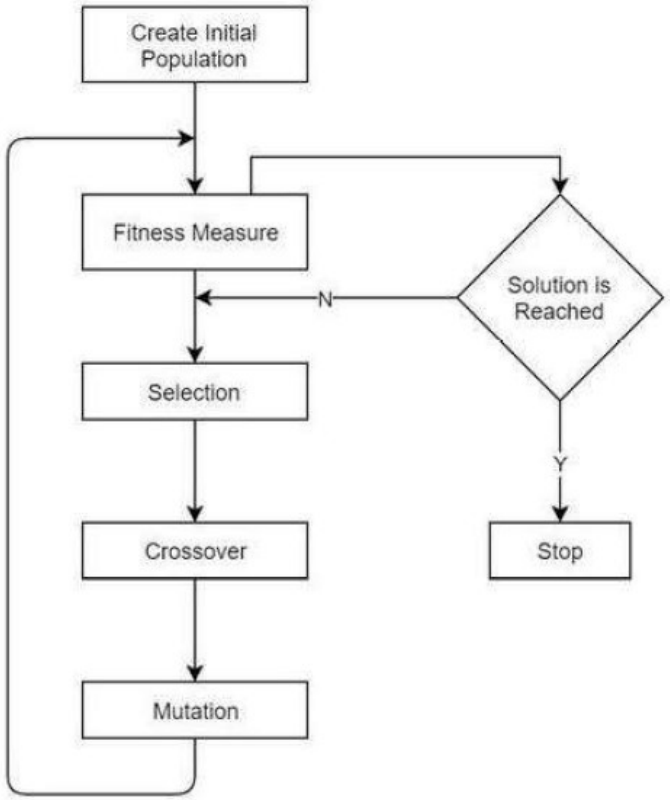
```
1.Enter candidate detail into database
2.Search for candidate
```

```
3.Probeble pairs of matching partners
4.Show all candidate details
0.Exit
****************************************
Enter your choice:4
------------------------------------------------------------------------
------------------------------------------------------------------------
Name : hiral Gender : female
Age : 22
Attribute[Qualification,Height,Weight,Color] : ["MCA","5.4","56","whitish"]
Preferece[Qualification,Height,Weight,Color] : ["MBA","5.9","65","whitish"]
```

LAB - 9

AIM: Implement Knapsack using genetic algorithms.

Solution:



Code:

Randomly initialize item and weight

```
import numpy as np
import pandas as pd
import random as rd
from random import randint
import matplotlib.pyplot as plt
item_number = np.arange(1,11)
weight = np.random.randint(1, 15, size = 10)
value = np.random.randint(10, 750, size = 10)
knapsack_threshold = 35    #Maximum weight that the bag of thief
```

```
The list is as :
Item No.    Weigl
1           5

2           8

3           7

4           8

5           6

6           7

7           3
```

Initial population

```
solutions_per_pop = 8
pop_size = (solutions_per_pop, item_number.shape[0])
print('Population size = {}'.format(pop_size))
initial_population = np.random.randint(2, size = pop_
initial_population = initial_population.astype(int)
num_generations = 50
print('Initial population: \n{}'.format(initial_popul
```

```
Population size = (8, 10)
Initial population:
[[1 0 1 0 1 1 0 1 0 1]
 [0 1 1 1 0 0 1 0 1 0]
```

Calculate Fitness and selection function

```
def cal_fitness(weight, value, population, threshold):
    fitness = np.empty(population.shape[0])
    for i in range(population.shape[0]):
        S1 = np.sum(population[i] * value)
        S2 = np.sum(population[i] * weight)
        if S2 <= threshold:
            fitness[i] = S1
        else :
            fitness[i] = 0
    return fitness.astype(int)
```

```
def selection(fitness, num_parents, population):
```

```
def crossover(parents, num_offsprings):
    offsprings = np.empty((num_offsprings, parents.shape[1]))
    crossover_point = int(parents.shape[1]/2)
    crossover_rate = 0.8
    i=0
    while (parents.shape[0] < num_offsprings):
        parent1_index = i%parents.shape[0]
        parent2_index = (i+1)%parents.shape[0]
        x = rd.random()
        if x > crossover_rate:
            continue
```

```
        parent2_index = (i+1)%parents.shape[0]
        offsprings[i,0:crossover_point] = parents[parent1_index,0:c
        offsprings[i,crossover_point:] = parents[parent2_index,cros
        i=+1
    return offsprings
```

```
def optimize(weight, value, population, pop_size, num_genera
    parameters, fitness_history = [], []
    num_parents = int(pop_size[0]/2)
    num_offsprings = pop_size[0] - num_parents
    for i in range(num_generations):
        fitness = cal_fitness(weight, value, population, thr
        fitness_history.append(fitness)
        parents = selection(fitness, num_parents, population
        offsprings = crossover(parents, num_offsprings)
        mutants = mutation(offsprings)
        population[0:parents.shape[0], :] = parents
        population[parents.shape[0]:, :] = mutants
```

```
    parameters.append(population[max_fitness[0][0],:])
    return parameters, fitness_history
```

```
parameters, fitness_history = optimize(weight, value, initial_population, pop_size, nu
print('The optimized parameters for the given inputs are: \n{}'.format(parameters))
selected_items = item_number * parameters
print('\nSelected items that will maximize the knapsack without breaking it:')
for i in range(selected_items.shape[1]):
  if selected_items[0][i] != 0:
    print('{}\n'.format(selected_items[0][i]))
```

```
Last generation:
[[0 1 1 1 1 0 1 0 0 0]
 [0 1 1 1 1 0 1 0 0 0]
 [0 1 1 1 1 0 1 0 0 0]
 [0 1 1 1 1 0 1 0 0 0]
 [0 1 1 1 1 0 1 0 0 0]
 [0 1 1 1 1 0 1 0 0 0]
 [0 1 1 1 1 0 1 0 0 0]
 [0 1 1 1 1 0 1 0 1 0]]

The optimized parameters for the given inputs are:
[array([0, 1, 1, 1, 1, 0, 1, 0, 0, 0])]

Selected items that will maximize the knapsack
2

3

4
```

LAB - 10

AIM: Travelling salesman problem using nearest neighbor heuristic and greedy edge heuristic.

1) Using Greedy edge Heuristic

```python
def checkForCycle(parent,edge):
    if parent[edge[1]]==parent[edge[2]]:
        if parent[edge[1]]==-1:
            parent[edge[1]]=parent[edge[2]]=edge[2]
            return False
        return True

    if parent[edge[1]]==-1:
        parent[edge[1]] = parent[edge[2]]
    elif parent[edge[2]]==-1:
        parent[edge[2]]=parent[edge[1]]
    else:
        n1,n2 = parent[edge[1]],parent[edge[2]]
        for i in range(len(parent)):
            if parent[i]==n1:
                parent[i] = n2
    return False
```

```python
def getWeight(edges,a,b):
    for w,x,y in edges:
        if x==a and y==b:
            return w
    print("Unexpected Error")
    exit(0)
```

```python
def getPathFromEdge(edges:list,sedge:list,n:int):
    cpath,deg = [],[0]*n
    parent = [-1]*n
    cpath.append([getWeight(edges,sedge[0],sedge[1]),sedge[0],sedge[1]])
    checkForCycle(parent,[0,sedge[0],sedge[1]])
    deg[sedge[0]],deg[sedge[1]] = 1,1
    for edge in edges:
        if deg[edge[1]]>=2 or deg[edge[2]]>=2:
            continue
        if checkForCycle(parent,edge):
            continue

        deg[edge[1]]+=1
        deg[edge[2]]+=1
        cpath.append(edge)
    oneD = []
    for i in range(n):
        if deg[i]==1:
            oneD.append(i)
    if len(oneD)!=2:
        print("Error")
        exit(0)
    cpath.append([getWeight(edges,oneD[0],oneD[1]),oneD[0],oneD[1]])
    return cpath
```

```python
def getNextEdge(cpath:List,cnode:int):
    for a,b,c in cpath:
        if b == cnode:
            cpath.remove([a,b,c])
            return c,a
        if c == cnode:
            cpath.remove([a,b,c])
            return b,a
    print("ERROR OCCURRED::getNextEdge()")
    exit(0)
```

```python
def printFormattedPath(cpath:list,n:int):
    path,cost = [cpath[0][1],cpath[0][2]],cpath[0][0]
    pnode = cpath[0][2]
    cpath.remove(cpath[0])

    for _ in range(n-1):
        pnode,wei = getNextEdge(cpath,pnode)
        cost+=wei
        path.append(pnode)
    print("Path: {","".join([str(path[i])+" -> " for i in range(n)]),path[n],"}")
    print("cost: ",cost)
```

```python
def rmain():
    n = int(input())
    data = [list(map(int,input().split())) for _ in range(n)]
    edges = []
    for i in range(n):
        for j in range(i+1,n):
            if data[i][j]>0:
                edges.append([data[i][j],i,j])
    edges.sort()
    cpath = getPathFromEdge(edges,[3,4],n)
    printFormattedPath(cpath,n)
    return

rmain()
```

Output:

```
C:\Users\NISARGI>python greedy.py
6
0 10 20 30 40 50
10 0 31 21 51 41
20 31 0 12 59 100
30 21 12 0 5 8
40 51 59 5 0 69
50 41 100 8 69 0
Path: { 3 -> 4 -> 2 -> 0 -> 1 -> 5 ->  3 }
cost:  143
```

2) **Using Nearest neighbor Heuristic:**

```
cpath,vis = [],[]

def getUtilPath(data:list,cpoint:int):
    global cpath,vis
    if 0 not in vis:
        return

    mni,mnval = -1,max(data[cpoint])
    for i in range(len(data[cpoint])):
        if mnval>data[cpoint][i] and data[cpoint][i]>0 and vis[i]==0:
            mni,mnval = i,data[cpoint][i]

    if mni==-1:
        print("Error Occurred")

    vis[mni] = 1
    cpath.append([mni,mnval])
    getUtilPath(data,mni)
    return
```

```
def printFormattedPaths(paths:list,data:list):
    for i in range(len(paths)):
        print("Path: Start Point:(",i+1,") :\n\t","".join([str(val[0]+1)+" - " for val in paths[i]]),i+1)
        cost = 0
        for j in range(len(paths[0])):
            cost+= paths[i][j][1]
        cost += data[paths[i][-1][0]][i]
        print("\tCost:",cost,"\n")
    return
```

```
  def rmain():
      n = int(input())
      data = [list(map(int,input().split())) for _ in range(n)]
      paths = getPaths(data)
      printFormattedPaths(paths,data)
      return

  rmain()
```

```
def getPaths(data):
    global cpath,vis
    paths = []
    for i in range(len(data)):
        vis,cpath = [0]*len(data),[]
        vis[i] = 1
        cpath.append([i,0])
        getUtilPath(data,i)
        paths.append(cpath)
    return paths
```

Output:

```
5
0 10 8 9 7
10 0 10 5 6
8 10 0 8 9
9 5 8 0 6
7 6 9 6 0
Path: Start Point:( 1 ) :
        1 - 5 - 2 - 4 - 3 -  1
       Cost: 34

Path: Start Point:( 2 ) :
        2 - 4 - 5 - 1 - 3 -  2
       Cost: 36

Path: Start Point:( 3 ) :
        3 - 1 - 5 - 2 - 4 -  3
       Cost: 34

Path: Start Point:( 4 ) :
        4 - 2 - 5 - 1 - 3 -  4
       Cost: 34

Path: Start Point:( 5 ) :
        5 - 2 - 4 - 3 - 1 -  5
       Cost: 34
```