LAB - 1

Title: Prolog program to find factorial of given number using recursive function.

Program:

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

Query (input & output)

1. What is the factorial of 5?

```
? - factorial(5, A)
    A = 120
```

2. What is the factorial of 3?

```
? - factorial(3, F).
    F = 6
```

Conclusion:

Hence from this lab, we learnt to find factorial using recursive function using prolog program.

LAB-2: Prolog program to generate fibonacci number using recursive function.

Program:

```
fibonacci(0,0).
fibonacci(1,1).
fibonacci(N,F):-
    N1 is N-1,
    N2 is N-2,
fibonacci(N1,F1),
fibonacci(N2,F2),
F1 is F1+F2.
```

Query (o/p & input)

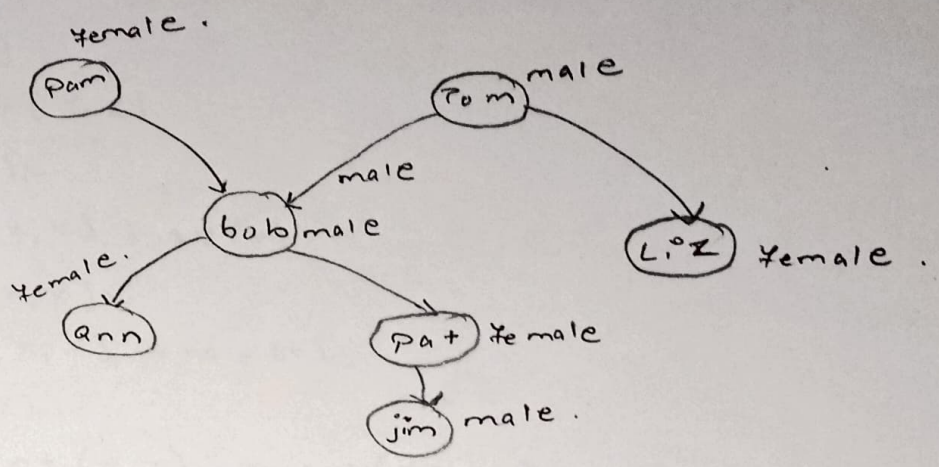
```
?- fibonacci(3,F).
2
```

{ o/p fibonacci की 3rd position (index) की value आएगी 0 1, 1, 2, 3, 5 }

Conclusion:-

Hence from this lab, we have learnt to find the fibonacci number using recursive function using prolog program.

LAB 3: Pro log program to represent following family tree.




Facts / KB

```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(pat, jim).
parent(bob, pat).

male(tom).
male(bob).
male(jim).
female(pam).
female(liz).
female(pat).

father(X, Y) :-
    parent(X, Y), male(X).
mother(X, Y) :-
    parent(X, Y), female(X).
grandparent(X, Y) :-
    parent(X, Z), parent(Z, Y).
grandmother(X, Y):
```

```
    grandparent (X, Y), female(X).
grandfather (X, Y) :-
    grandparent (X, Y), male(X).
child (X, Y) :-
    parent (Y, X).
son (X, Y) :-
    child (X, Y), male (X).
daughter (X, Y) :-
    child (X, Y), female (X).
wife (X, Y) :-
    parent (X, Z), parent (Y, Z), female (X).
husband (X, Y) :-
    parent (X, Z), parent (Y, Z), male (X).
sister (X, Y) :-
    parent (Z, X), parent (Z, Y), female (X).
brother (X, Y) :-
    parent (Z, X), parent (Z, Y), male (X).
```

Query

```
?- parent (X, Y).
X = pam,
Y = bob;
X = tom,
Y = bob;
X = tom,
Y = liz;
X = bob,
Y = ann;
X = pat,
```

Y = jim ;

X = bob,

Y = pat.

? - sister (Y, bob).

Y = liz ;

? - brother ( X, liz ).

X = bob.

? - grandparent ( X, jim ).

X = bob.

Conclusion:-

This is the third lab of Artificial Intelligence. In this lab, we have done prolong program to represent family tree.

LAB - 04

For given statement write a prolog program & answer the queries.

1. Jia is a woman.
2. John is a man
3. John is healthy
4. Jia is healthy.
5. John is wealthy.
6. Anyone is a traveller if he is wealthy and healthy
7. Anyone can travel if he is a traveller.

Tasks

- Convert those statement into facts / predicate.

Query (Input & output)

1. Who can travel?
2. Who is healthy and wealthy?
3. Who is healthy?
4. Jia is man.

Facts

```
women(jia).
man(John).
healthy(John).
healthy ~~(John)~~ (jia).
wealthy (John).
traveller(X):-
        wealthy(X), healthy(X).
travel(X):-
        traveller(X).
```

Query.

~~1. ?- travel~~

1. Who can travel?

?- travel(x)

x = john

2. Who is healthy and wealthy?

?- healthy(x), wealthy(x).

x = john

3. Who is healthy?

?- healthy(x).

x = john ;

x = jia

4. Jia is man

?- man(jia).

false

Conclusion:-

This is the fourth lab of AI. we have compiled this predicate using swi-prolog.

LAB-5

Title: Map Coloring with prolog (CSP)

Theory:

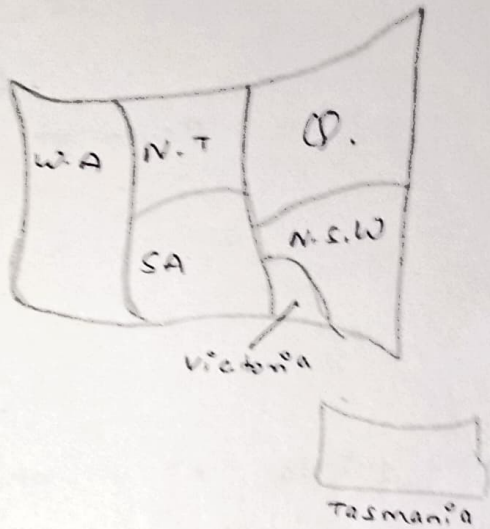
Constraint Satisfaction Problem:

A constraint satisfaction Problem (CSP) requires a value, selected from a given finite domain, to be assigned to each variable in the problem, so that all constraints relating the variables are satisfied. Many combinatorial problems in operational research, such as scheduling & timetabling, can be formulated as CSPs.

Program:

```
adjacent(wa, nt).
adjacent(wa, sa).
adjacent(nt, q).
adjacent(nt, sa).
adjacent(q, nsw).
adjacent(nsw, sa).
adjacent(nsw, v).
adjacent(sa, nt).
adjacent(sa, v).
adjacent(nt, wa).
adjacent(nsw, q).
adjacent(sa, wa).
adjacent(q, nt).
adjacent(sa, q).
adjacent(sa, nsw).
adjacent(v, nsw).
adjacent(v, sa).

color(wa, red, a)
color(nt, green, a)
color(q, red, a)
color(nsw, green, a)
color(v, red, a)
```



```
color (sa, blue, a)
color (wa, red, b)
color (nt, red, b)
color (q, blue, b)
color (nsw, red, b)
color (v, blue, b)
color (sa, green, b)
color (t, blue, a)
color (t, green, b)

conflict (X, Y, Z) :-
    adjacent (x, y), color (X, C, Z),
                ~~color (y, c, z).~~
                color (Y, C, Z).
```

Query (Input and Output)

1. Which state is adjacent to nsw?

```
? - adjacent (nsw, x).
x = sa ;
x = sa ;
x = v ;
x = q.
```

2. Which color is assigned to v in coloring scheme b?

```
? - color (v, C, b).
C = blue.
```

3. Has wa same color with nt in coloring scheme 'b'?

```
? - conflict (wa, nt, b).
true.
```

4. Has sa different color with v in coloring scheme 'a'?

```
? - conflict (sa, v, a).
false.
```

5. which state is adjacent to WA?

?- adjacent(wa, x).

x = nt;

x = sa.

Conclusion:-

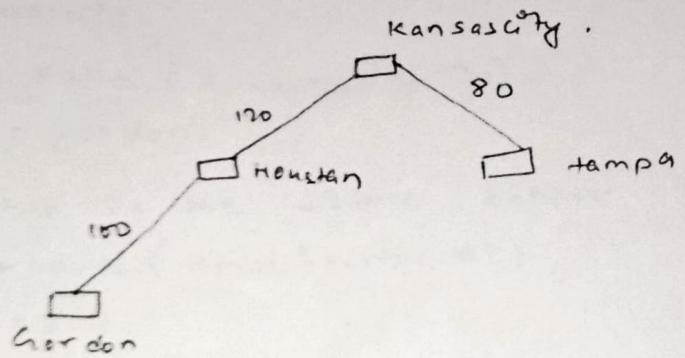
This is the fifth lab of Artificial Intelligent. In this lab we have successfully performed the example of constraint satisfaction problem i.e Map Coloring ~~with~~ ~~Prolog~~ with different queries.

Lab - 6

**Title:** Prolog Program to solve travelling salesman program.

**Objective:**

The traveling salesman problem (TSP) is an algorithm problem tasked with finding the shortest possible route to multiple destinations and returning the starting point.




**Fact:**

```
distance (gordon, houston, 100).
distance (houston, gordon, 100).
distance (houston, kansascity, 120).
distance (kansascity, houston, 120).
distance (tampa, kansascity, 80).
distance (kansascity, tampa, 80).

route (X, Y, Z) :-
    distance (X, Y, Z).

route (X, Y, Z) :-
    distance (X, W, D1),
    distance (W, Y, D2),
    X \= Y,
    Z is D1 + D2.
```

```
route (X, Y, Z) :-
    distance (X, W, D1),
    distance (W, P, D2),
    distance (P, Y, D3),
    X \= P,
    W \= Y,
    Z is D1 + D2 + D3.
```

Query:

1. Which city is located at distance of 270 from Kansas city.

```
?- route (X, kansascity, 220).
X = gordon.
```

2. What is the distance between tampa & houston?

```
?- route (tampa, houston, D).
D = 200
```

3. Is 80 distance between gordon & Houston?

```
?- route (gordon, houston, 80).
false
```

Conclusion:-

This is the sixth lab of AI written in SWI-prolog. In this lab, we have find out the shortest distance between the two cities.