* **Write a PROLOG program to implement the family tree and demonstrate the family relationship.**

**CODE :**

% gender

male(john).

male(paul).

male(david).

male(mark).

female(lisa).

female(mary).

female(anna).

female(susan).

% parent(child, parent)

parent(paul, john).

parent(paul, lisa).

parent(mary, john).

parent(mary, lisa).

parent(anna, david).

parent(anna, mary).

parent(mark, david).

parent(mark, mary).

% --- Rules ---

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

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

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

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

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

brother(X, Y) :- sibling(X, Y), male(X).

sister(X, Y) :- sibling(X, Y), female(X).

grandparent(X, Y) :- parent(X, Z), parent(Z, Y).

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

aunt(X, Y) :- parent(X, Z), cousin(Z, Y), female(X).

cousin(X, Y) :- parent(A, X), parent(B, Y), sibling(A, B).

ancestor(X, Y) :- parent(X, Y).

ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).

**OUTPUT:**

| ?- father(paul, X).

X = john ?

yes

| ?- mother(mary, X).

X = john ?

yes

| ?- brother(john, X).

X = lisa ?

yes

| ?- sister(lisa , X).

X = john ?

yes

| ?- sibling(lisa, X).

X = john ?

yes

| ?- parent(X, paul).

no

| ?- grandparent(anna,X).

X = john ?

yes

| ?- uncle(paul,X).

no

| ?- aunt(mary,X).

no

| ?- cousin(anna,X).

no

| ?- ancestor(anna,X).

X = david ? ;

X = mary ? ;

X = john ?

(16 ms) yes

| ?-

* **Write a PROLOG program to implement conc(L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.**

**CODE :**

% conc(L1,L2,L3) where L3 is L2 appended to L1

go :- write('Enter a list : '),read(L1),nl,

write('Enter the list to be appended : '),read(L2),nl,

conc(L1,L2,L3),

write('The new list is : '),write(L3).

conc([],L,L).

conc([H|T1],L2,[H|T3]) :- conc(T1,L2,T3).

**OUTPUT:**

* **Write a PROLOG program to implement reverse(L, R) where List L is original and List R is reversed list.**

**CODE :**

% Reversing a list reverse(L,R)

go :- write('Enter the List : '),read(L),nl,

myreverse(L,R),

write('The reversed list is '),write(R).

myreverse([],[]).

myreverse([H],[H]).

myreverse([H|T],R) :- myreverse(T,R1),conc(R1,[H],R).

conc([],L1,L1).

conc([H|T],L2,[H|L3]) :- conc(T,L2,L3).

OUTPUT:



* **Write a PROLOG program to calculate the sum of two numbers.**

**CODE :**

% Addition of two number

go :- write('Enter the first number'),nl,read(A),

write('Enter the second the number'),nl,read(B),

sum(A,B,C),

write('The Sum is'),nl,

write(C).

sum(X,Y,Z):- Z is X+Y.

**OUTPUT:**



* **Write a PROLOG program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.**

**CODE :**

% Maximum of two numbers

go :- write('Enter the first number : '),read(X),nl,

write('Enter the second number : '),read(Y),nl,

max(X,Y,M),

write('Maximum of the two numbers is : '),write(M).

max(X,Y,M) :- X>=Y,

M is X.

max(X,Y,M) :- X<Y,

M is Y.

**OUTPUT:**



* **Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N.**

**CODE :**

% Implement factorial (N,F) where

fact(0,1).

fact(N,F) :- N>0,

N1 is N-1,

fact(N1,F1),

F is N\*F1.

**OUTPUT:**



* **Write a program in PROLOG to implement generate\_fib(N,T) where T represents the Nth term of the Fibonacci series.**

**CODE :**

% PROLOG to implement generate\_fib(N,T)

fibo(1,0):- !.

fibo(2,1):- !.

fibo(3,1):- !.

fibo(A,X):- M is A-1,

N is A-2,

fibo(M,W),

fibo(N,L),X is W+L.

**OUTPUT:**



* **Write a PROLOG program to implement power (Num, Pow, Ans) : where Num is raised to the power Pow to get Ans.**

**CODE :**

% power(Num, Pow, Ans)

power(Num,0,1) :- !.

power(Num,Pow,Ans):-

Pow > 0,

Pow1 is Pow - 1,

power(Num,Pow1,Temp),

Ans is Temp \* Num.

**OUTPUT:**

* **PROLOG program to implement multi (N1, N2, R) : where N1 and N2 denotes the numbers to be multiplied and R represents the result.**

**CODE :**

% multi(N1,N2,R) : where R is N1\*N2

multi(N1,N2,R):-

R is N1 \* N2.

**OUTPUT:**

* **Write a PROLOG program to implement memb(X, L): to check whether X is a member of L or not.**

**CODE :**

% Member function to check if X is a member of L.

go :- write('Enter the element : '),read(X),nl,

write('Enter the list : '),read(L),

mem(X,L).

mem(X,[X|\_]).

mem(X,[H|T]):- mem(X,T).

**OUTPUT:**



* **Write a PROLOG program to implement sumlist(L, S) so that S is the sum of a given list L.**

**CODE :**

% Sum of a list sumlist(L,S) S is sum of L

go :- write('Enter the list : '),read(L),nl,

sumlist(L,S),

write('The sum of the list is : '),write(S).

sumlist([],0).

sumlist([H|T],S) :- sumlist(T,R),

S is R+H.

**OUTPUT:**

* **Write a PROLOG program to implement two predicates evenlength(List) and oddlength(List) so that they are true if their argument is a list of even or odd length respectively.**

**CODE :**

% Main program

go :-

write('Enter a list: '), read(L), nl,

( evenlength(L) ->

write('The list is of even length.')

; oddlength(L) ->

write('The list is of odd length.')

;

write('Invalid input.')

).

% evenlength: true if list has even number of elements

evenlength([]).

evenlength([\_,\_|T]) :- evenlength(T).

% oddlength: true if list has odd number of elements

oddlength([\_]).

oddlength([\_,\_|T]) :- oddlength(T).

**OUTPUT:**



* **Write a PROLOG program to implement maxlist(L, M) so that M is the maximum number in the list.**

**CODE :**

go :- write('Enter a list : '),read(L),nl,

maxlist(L,M),

write('The maximum element in the list is : '),write(M).

maxlist([X],X) :- !.

maxlist([H1,H2|T],M) :- maxtwo(H1,H2,M1),

maxlist([M1|T],M).

maxtwo(X,Y,M) :- X>=Y,

M is X.

maxtwo(X,Y,M) :- X<Y,

M is Y.

**OUTPUT:**



* **Write a PROLOG program to implement insert(I, N, L, R) that inserts an item I into Nth position of list L to generate a list R.**

**CODE :**

% Insertion at index in list

go :- write('Enter the list : '),read(L),

write('Enter the position : '),read(N),

write('Enter to be inserted : '),read(I),

insert\_nth(I,N,L,R),

write('The resultant list is : '),write(R).

insert\_nth(I,1,T,[I|T]) :- !.

insert\_nth(X,N,[Y|T],[Y|T1]) :- N>1,

N1 is N-1,

insert\_nth(X,N1,T,T1).

**OUTPUT:**



* **Write a PROLOG program to implement delete(N, L, R) that removes the element on Nth position from a list L to generate a list R.**

**CODE :**

% Deleting the nth element in a list

go :- write('Enter the list : '),read(L),nl,

write('Enter the position : '),read(N),nl,

delete\_nth(N,L,R),

write('The new list is : '),write(R).

delete\_nth(1,[\_|T],T).

delete\_nth(N,[H|T1],[H|T2]) :- N>1,

N1 is N-1,

delete\_nth(N1,T1,T2).

**OUTPUT:**

