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

**Code:**

parent(sohan,priya).

parent(priya,bobby).

parent(tilak,bobby).

parent(tilak,lalita).

parent(bobby,anita).

parent(bobby,pratiksha).

parent(pratiksha,john).

male(sohan).

male(tilak).

male(bobby).

male(john).

female(priya).

female(lalita).

female(anita).

female(pratiksha).

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

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

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

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

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

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

grandchild(X,Z):- child(X,Y), child(Y,Z).

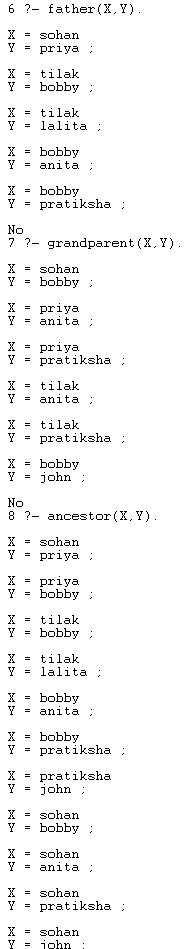
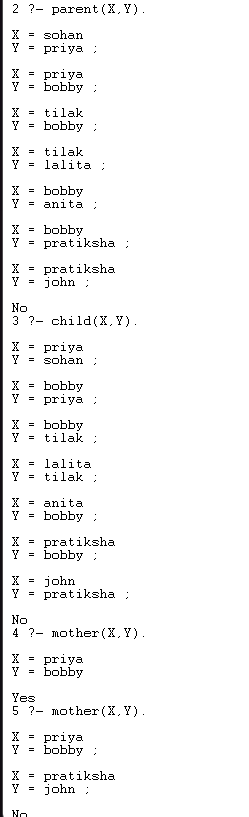
grandfather(X,Z):-parent(X,Y),parent(Y,Z), male(X).

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

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

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

**Output:**



**2. 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([],L,L).

conc([X|L1],L2,[X|L3]):- conc(L1,L2,L3).

**Output:**



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

**Code:**

reverse([],[]).

reverse([H|T],L):- reverse(T,R), append(R,[H],L).

**Output:**



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

**Code:**

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

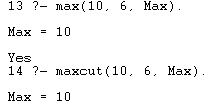
**Code:**

max(X, Y, Max):- X>=Y, Max is X.

max(X, Y, Max):- X<Y, Max is Y.

maxcut(X, Y, Max):- X>=Y, !, Max=X; Max=Y.

**Output:**



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

**Code:**

factorial:-write('Enter number: '),read(N),

factorial(N,F),

write('The factorial of '),write(N),

write(' is '),write(F).

factorial(0,1).

factorial(N,F):- N>0, N1 is N-1,

factorial(N1, F1),

F is N\*F1.

**Output:**



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

**Code:**

generate\_fib(0,0).

generate\_fib(1,1).

fibonacci:-write('Enter number: '),read(N),generate\_fib(N,T),

write('The '), write(N), write('th index fibonacci number is '), write(T).

generate\_fib(N,T):-

N>1,

N1 is N-1,

N2 is N-2,

generate\_fib(N1, T1),

generate\_fib(N2, T2),

T is T1 + T2.

**Output:**



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

**Code:**

start:-write('Enter the number: '), read(N),

write('Enter the power: '), read(P),

power(N,P,R), write(N), write(' to the power '), write(P), write(' is '), write(R).

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

power(N,P,R):- P1 is P-1, power(N,P1,R1), R is R1\*N.

**Output:**



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

**Code:**

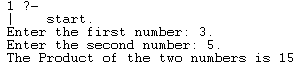
start:-write('Enter the first number: '), read(N1),

write('Enter the second number: '), read(N2),

product(N1,N2,R),write('The Product of the two numbers is '), write(R).

product(N1,N2,R):- R is N1\*N2.

**Output:**



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

**Code:**

member(X,[X|Tail]).

member(X,[Head|Tail]):- member(X,Tail).

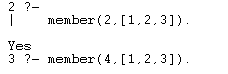
memb(X,[X|T]):-!.

memb(X,[Y|T]):- memb(X,T).

memb1(X,[X|T]).

memb1(X,[Y|T]):- memb1(X,T).

**Output:**



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

**Code:**

sumlist([],0).

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

sumlist(T,R1),

R is H+R1.

**Output:**



**12. 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:**

evenlength([]).

evenlength([First|Rest]):- oddlength(Rest).

oddlength([First|Rest]):- evenlength(Rest).

/\*

listlength([], 0).

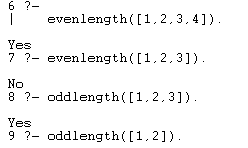
listlength([H|T], N):- listlength(T,N1),N is N1+1.

oddlength(List):-listlength(List,N),N1 is N mod 2, N1=1.

evenlength(List):-listlength(List,N),N1 is N mod 2, N1=0.

\*/

**Output:**



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

**Code:**

/\*

max\_list([H|T],Max):- max\_list(T, H, Max).

max\_list([],Max,Max).

max\_list([H|L1],Max0,Max):- Max0 >= H, max\_list(L1, Max0, Max).

max\_list([H|L1],Max0,Max):- Max0 < H, max\_list(L1, H, Max).

\*/

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

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

max\_list([X], X).

max\_list([X|Tail],Max):- max\_list(Tail,Maxtail), max(X,Maxtail,Max).

**Output:**



**14. 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:**

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

write('Enter the element to be inserted:- '),read(X),

write('Enter the position to inserted:- '),read(N),

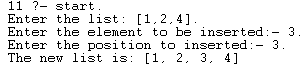
insert\_nth(X,N,L,R),

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

insert\_nth(X,1,L,[X|L]).

insert\_nth(X,N,[H|L],[H|R]):- N>1,N1 is N-1,insert\_nth(X,N1,L,R).

**Output:**



**15. 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:**

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

write('Enter the element to be deleted:- '),read(X),

del(X,L,R),

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

del(X,[X|Tail],Tail).

del(X,[Y|Tail],[Y|Tail1]):- del(X,Tail,Tail1).

**Output:**

