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

sum(X,Y,Z) :-

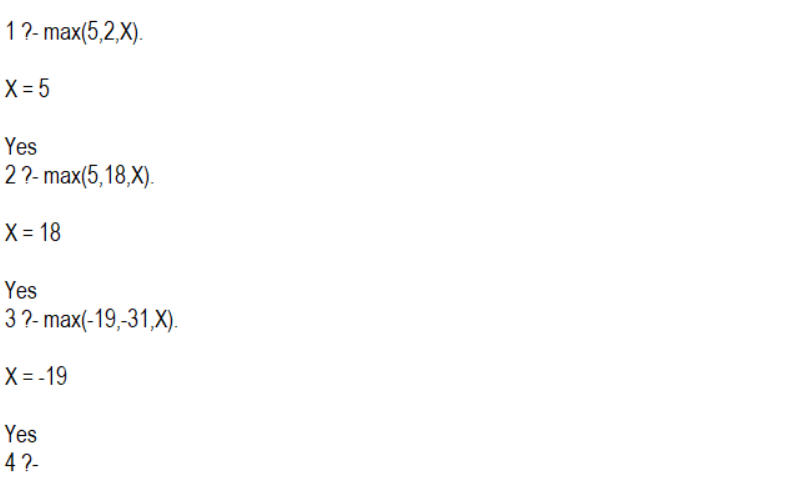
   Z is X+Y.



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

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

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



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

factorial(0,1).

factorial(N,F) :-

    N > 0,

    N1 is N - 1,

    factorial(N1,F1),

    F is N \* F1.



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

generate\_fib(1, 0) :- !.

generate\_fib(2, 1) :- !.

generate\_fib(N, T) :-

    N > 2,

    N1 is N - 1,

    N2 is N - 2,

    generate\_fib(N1, T1),

    generate\_fib(N2, T2),

    T is T1 + T2.



**5. Write a Prolog program to implement GCD of two numbers.**

gcd(X,X,X).

gcd(0,X,X).

gcd(X,0,X).

gcd(X,Y,G) :-

    X > Y,

    X1 is X - Y,

    gcd(X1,Y,G).

gcd(X,Y,G) :-

    X < Y,

    Y1 is Y - X,

    gcd(X,Y1,G).



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

power(Num,0,1).

power(Num,Pow,Ans) :-

    Pow > 0,

    Pow1 is Pow - 1,

    power(Num,Pow1,Ans1),

    Ans is Num \* Ans1.

power(Num,Pow,Ans) :-

    Pow < 0,

    Pow1 is Pow + 1,

    power(Num,Pow1,Ans1),

    Ans is Ans1 / Num.



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

multi(N1, N2, R) :-

    R is N1 \* N2.

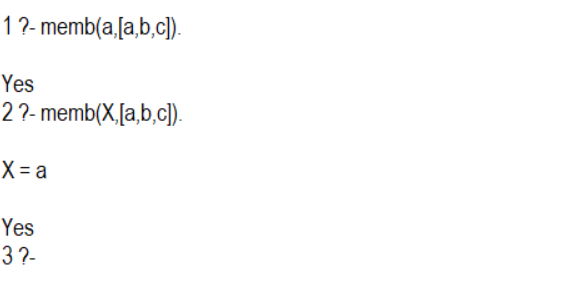


**8. Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not.**

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

memb(X, [\_|T]) :-

    memb(X, T).

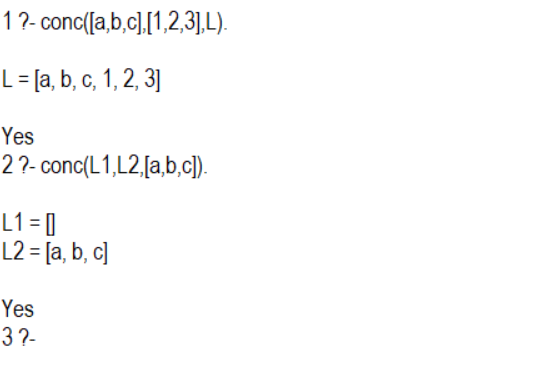


**9. Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.**

conc([], L, L).

conc([H|T], L, [H|L1]) :-

    conc(T, L, L1).



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

conc([], L, L).

conc([H|T], L, [H|L1]) :-

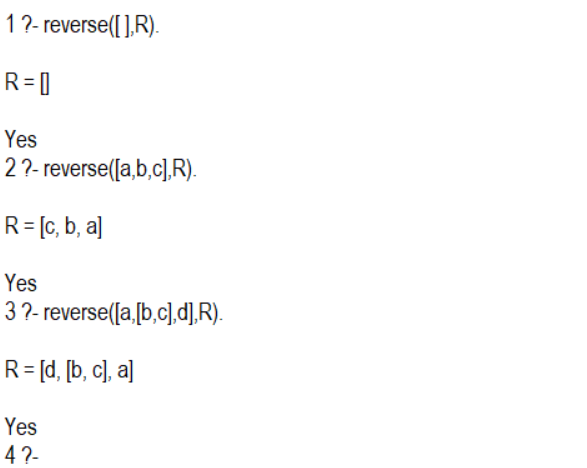
    conc(T, L, L1).

reverse([], []).

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

    reverse(T, R1),

    conc(R1, [H], R).



**11. Write a program in PROLOG to implement palindrome (L) which checks whether a list L is a palindrome or not.**

conc([], L, L).

conc([H|T], L, [H|L1]) :-

    conc(T, L, L1).

reverse([], []).

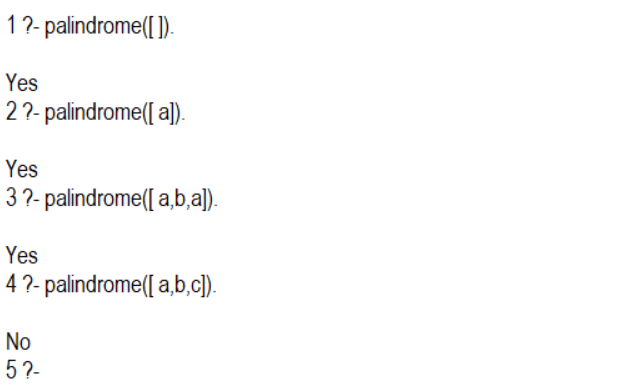
reverse([H|T], R) :-

    reverse(T, R1),

    conc(R1, [H], R).

palindrome(L) :-

    reverse(L, L).



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

sumlist([], 0).

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

    sumlist(T, S1),

    S is S1 + H.



**13. 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.**

evenlength([]).

evenlength([\_|T]) :-

    oddlength(T).

oddlength([\_|T]) :-

    evenlength(T).



**14. Write a Prolog program to implement nth\_element (N, L, X) where N is the desired position, L is a list and X represents the Nth element of L.**

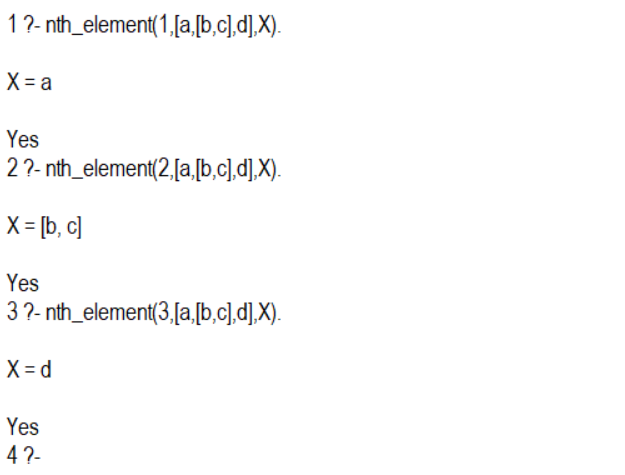
nth\_element(1, [H|\_], H).

nth\_element(N, [\_|T], X) :-

    N > 1,

    N1 is N - 1,

    nth\_element(N1, T, X).



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

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

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

maxlist([X],X).

maxlist([H|T],M):-maxlist(T,M1),M is max(H,M1).



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

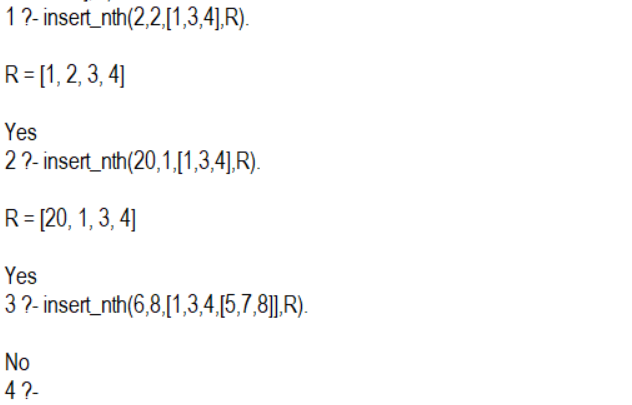
insert\_nth(I, 1, L, [I|L]).

insert\_nth(I, N, [H|T], [H|R]) :-

     N > 1,

     N1 is N-1,

    insert\_nth(I, N1, T, R).



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

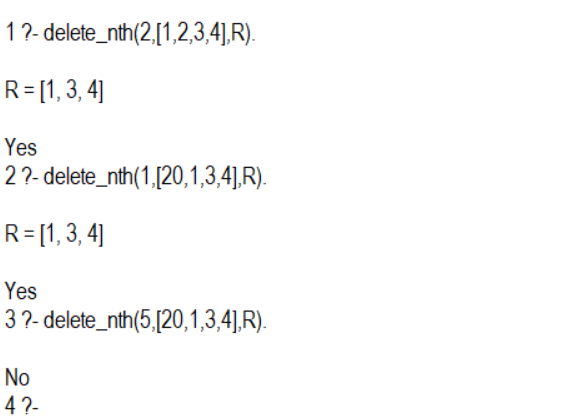
delete\_nth(1, [\_|T], T).

delete\_nth(N, [H|T], [H|R]) :-

     N > 1,

     N1 is N-1,

    delete\_nth(N1, T, R).



**18. Write a program in PROLOG to implement merge (L1, L2, L3) where L1 is first ordered list and L2 is second ordered list and L3 represents the merged list.**

merge([],L,L).

merge(L,[],L).

merge([H1|T1],[H2|T2],[H1|T3]) :-

   H1 =< H2,

   merge(T1,[H2|T2],T3).

merge([H1|T1],[H2|T2],[H2|T3]) :-

   H1 > H2,

   merge([H1|T1],T2,T3).

