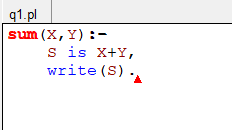
Q1. Write a prolog program to calculate the sum of two numbers.

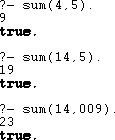
**CODE**

sum(X,Y):- S is X+Y,

write(S).



**OUTPUT**



Q2. Write a Prolog program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.

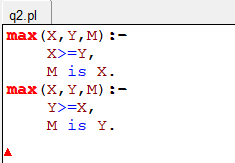
**CODE**

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

M is X.

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

M is Y.



**OUTPUT**



Q3. Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N.

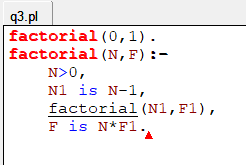
**CODE**

factorial(0,1). factorial(N,F):-

N>0,

N1 is N-1,

factorial(N1,F1), F is N\*F1.



**OUTPUT**



Q4. Write a program in PROLOG to implement generate\_fib(N,T) where T represents the Nth term of the fibonacci series.

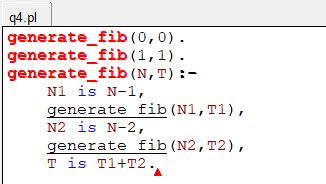
**CODE**

generate\_fib(0,0). generate\_fib(1,1). generate\_fib(N,T):-

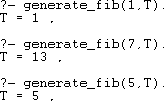
N1 is N-1,

generate\_fib(N1,T1), N2 is N-2,

generate\_fib(N2,T2), T is T1+T2.



**OUTPUT**



Q5. Write a Prolog program to implement GCD of two numbers.

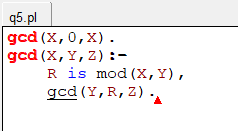
**CODE**

gcd(X,0,X).

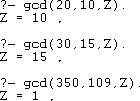
gcd(X,Y,Z):-

R is mod(X,Y),

gcd(Y,R,Z).



**OUTPUT**



Q6. Write a Prolog program to implement power (Num,Pow, Ans) : where Num is raised to the power Pow to get Ans.

**CODE**

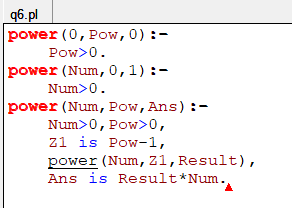
power(0,Pow,0):- Pow>0.

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

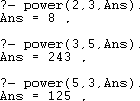
power(Num,Pow,Ans):- Num>0,Pow>0,

Z1 is Pow-1,

power(Num,Z1,Result), Ans is Result\*Num.



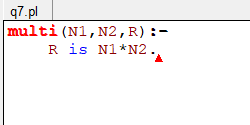
**OUTPUT**



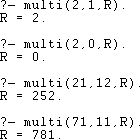
Q7. Prolog program to implement multi (N1, N2, R): where N1 and N2 denotes the numbers to be multiplied and R represents the result.

**CODE**

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



**OUTPUT**

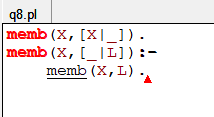


Q8. Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not

**CODE**

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

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



**OUTPUT**

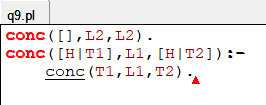


Q9. Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.

**CODE**

conc([],L2,L2). conc([H|T1],L1,[H|T2]):-

conc(T1,L1,T2).



**OUTPUT**



Q10. Write a Prolog program to implement reverse (L, R) where List L is original and List R is reversed list.

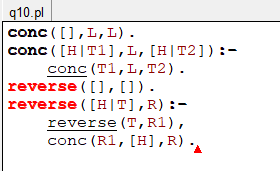
**CODE**

conc([],L,L). conc([H|T1],L,[H|T2]):-

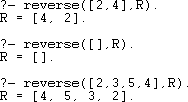
conc(T1,L,T2).

reverse([],[]).

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



**OUTPUT**



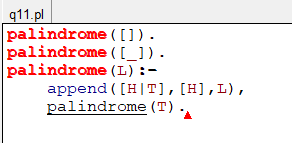
Q11. Write a program in PROLOG to implement palindrome (L) which checks whether a list L is a palindrome or not.

**CODE**

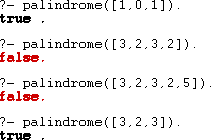
palindrome([]).

palindrome([\_]). palindrome(L):-

append([H|T],[H],L), palindrome(T).



**OUTPUT**

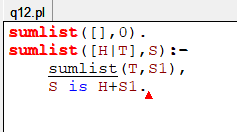


Q12. Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.

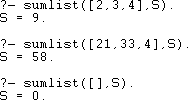
**CODE**

sumlist([],0).

sumlist([H|T],S):- sumlist(T,S1), S is H+S1.



**OUTPUT**



Q13. 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.

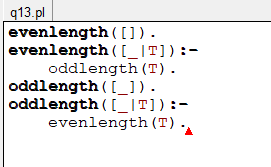
**CODE**

evenlength([]).

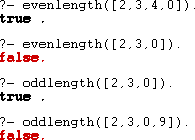
evenlength([\_|T]):- oddlength(T).

oddlength([\_]).

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



**OUTPUT**



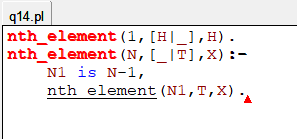
Q14. Write a Prolog program to implement nth\_element (N, L, X) where N is the desired position, L is a list and X represents the Nth element of L.

**CODE**

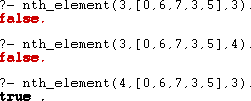
nth\_element(1,[H|\_],H).

nth\_element(N,[\_|T],X):- N1 is N-1,

nth\_element(N1,T,X).



**OUTPUT**



Q15. Write a Prolog program to implement maxlist(L, M) so that M is the maximum number in the list.

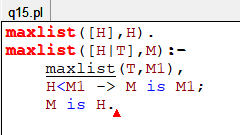
**CODE**

maxlist([H],H).

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

H<M1 -> M is M1;

M is H.



**OUTPUT**



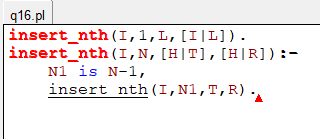
Q16. Write a prolog program to implement insert\_nth (I, N, L, R) that inserts an item I into Nth position of list L to generate a list R.

**CODE**

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

insert\_nth(I,N,[H|T],[H|R]):- N1 is N-1,

insert\_nth(I,N1,T,R).



**OUTPUT**



Q17. 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.

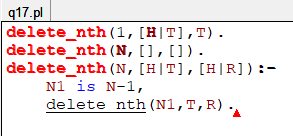
**CODE**

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

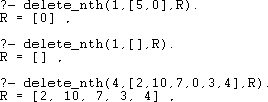
delete\_nth(N,[],[]).

delete\_nth(N,[H|T],[H|R]):- N1 is N-1,

delete\_nth(N1,T,R).



**OUTPUT**



Q18. 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.

**CODE**

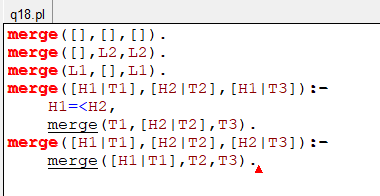
merge([],[],[]). merge([],L2,L2).

merge(L1,[],L1).

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

merge(T1,[H2|T2],T3).

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



**OUTPUT**

