

Question 1

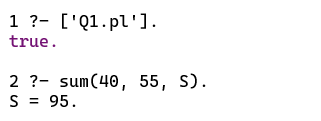
# Write a prolog program to calculate the sum of two numbers.

CODE

sum (X, Y, Z) :-

Z is X + Y.

OUTPUT



Question 2

# 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,

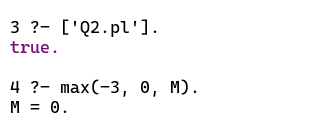
M is X.

max (X, Y, M) :-

Y >= X,

M is Y.

OUTPUT



Question 3

# 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) :-

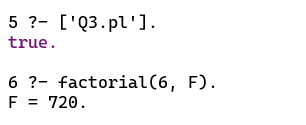
N1 is N - 1,

factorial (N1, Y),

F is Y \* N,

!.

OUTPUT



Question 4

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

CODE

generate\_fib (1, 0).

generate\_fib (2, 1).

generate\_fib (N, T) :-

N1 is N - 1,

N2 is N - 2,

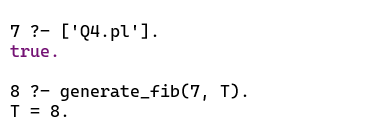
generate\_fib (N1, T1),

generate\_fib (N2, T2),

T is T1 + T2,

!.

OUTPUT



Question 5

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

CODE

gcd (X, X, X).

gcd (0, X, X) :- !.

gcd (X, 0, X) :- !.

gcd (X, Y, D) :-

X < Y,

Y1 is Y - X,

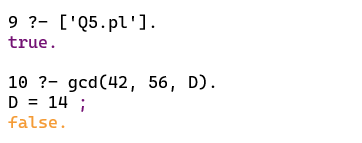
gcd (X, Y1, D).

gcd (X, Y, D) :-

Y < X,

gcd (Y, X, D).

OUTPUT



Question 6

# Write a prolog program to implement power(Num, Pow, Ans) where Num is raised to the power Pow to get Ans.

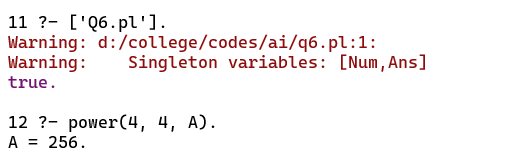
CODE

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

power (Num, Pow, Ans) :-

Ans is Num ^ Pow.

OUTPUT



Question 7

# Write a prolog program to implement multi(N1, N2, R) where N1 and N2 denotes the numbers to be multiplied and R represents the result.

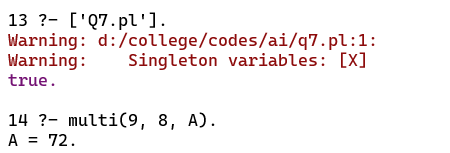
CODE

multi (X, 0).

multi (N1, N2, R) :-

R is N1 \* N2.

OUTPUT



Question 8

# Write a prolog program to implement memb(X, L) to check whether X is a member of L or not.

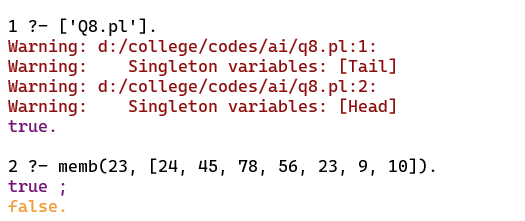
CODE

memb (X, [X|Tail]).

memb (X, [Head|Tail]) :-

memb (X,Tail).

OUTPUT



Question 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.

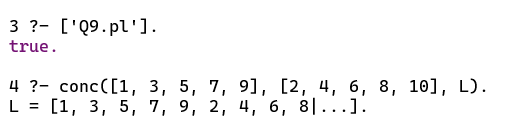
CODE

conc ([], L, L).

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

conc (L1, L2, L3).

OUTPUT



Question 10

# Write a prolog program to implement reverse(L, R) where list L is original and list R is reversed list.

CODE

append ([], L, L).

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

append (L1, L2, L3).

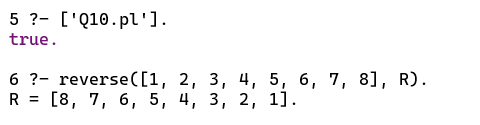
reverse ([], []).

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

reverse (T, L1),

append (L1, [H], R).

OUTPUT



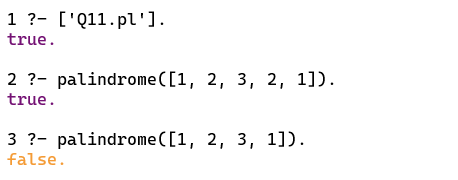
Question 11

# Write a prolog program to implement palindrome(L) which checks whether a list L is a palindrome or not.

CODE

palindrome(L) :- reverse(L, L).

OUTPUT



Question 12

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

CODE

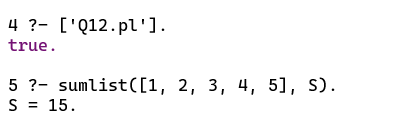
sumlist ([], 0).

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

sumlist (T, S1),

S is H + S1.

OUTPUT



Question 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.

CODE

evenlength ([]).

evenlength ([\_|[\_|List]]) :-

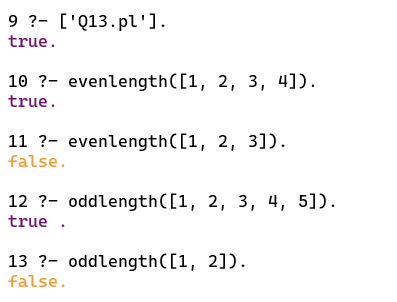
evenlength (List).

oddlength ([\_]).

oddlength ([\_|[\_|List]]) :-

oddlength (List).

OUTPUT



Question 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.

CODE

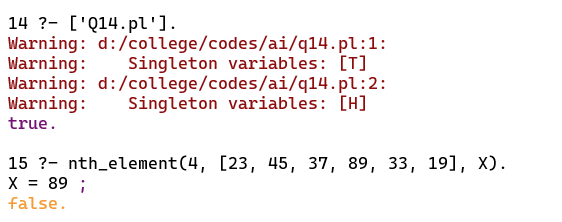
nth\_element (1, [H|T], H).

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

N1 is N-1,

nth\_element (N1, T, X).

OUTPUT



Question 15

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

CODE

max (X, Y, Z) :-

X > Y,

Z is X.

max (X, Y, Z) :-

X =< Y,

Z is Y.

maxlist ([], 0) :- !.

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

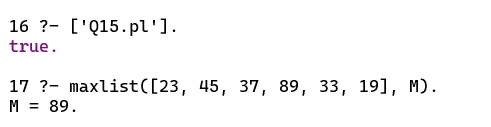
maxlist ([H|T], R) :-

maxlist (T, R1),

max (H, R1, R),

!.

OUTPUT



Question 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.

CODE

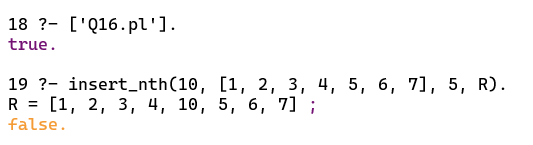
insert\_nth (Item, 1, List, [Item|List]).

insert\_nth (Item, Pos, [H|List], [H|Result]) :-

Pos1 is Pos - 1,

insert\_nth (Item, Pos1, List, Result).

OUTPUT



Question 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.

CODE

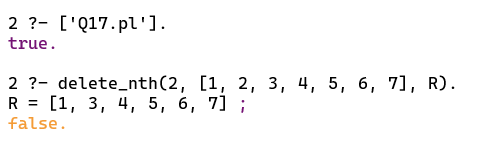
delete\_nth (1, [\_|List], List).

delete\_nth (Pos, [H|List], [H|Result]) :-

Pos1 is Pos - 1,

delete\_nth (Pos1, List, Result).

OUTPUT



Question 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.

CODE

merge (X, [], X).

merge ([], Y, Y).

merge ([X|X1], [Y|Y1], [X|Z]) :-

X < Y,

!,

merge (X1, [Y|Y1], Z).

merge ([X|X1], [Y|Y1], [X,Y|Z]) :-

X = Y,

!,

merge (X1, Y1, Z).

merge ([X|X1], [Y|Y1], [Y|Z]) :-

X > Y,

!,

merge ([X|X1], Y1, Z).

OUTPUT

