

DEEN DAYAL UPADHYAYA COLLEGE

UNIVERSITY OF DELHI

**PRACTICAL FILE**

**ARTIFICIAL INTELLIGENCE**

SUBMITTED BY- SUBMITTED TO -

NAME – AADITYA SUMAN DR. ANUJA SONI

ROLL NO – 21HCS4101

SEM -VI

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

**CODE**

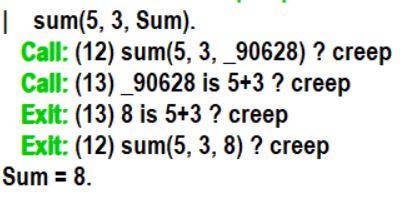
sum(X, Y, Sum) :-

Sum is X + Y.

**OUTPUT**

****

OUTPUT WITH TRACE:



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, X) :-

X >= Y.

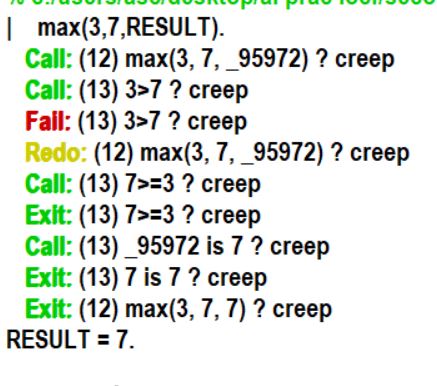
max(X, Y, Y) :-

X < Y.

**OUTPUT**



OUTPUT WITH TRACE:



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) :-

N > 0,

N1 is N - 1,

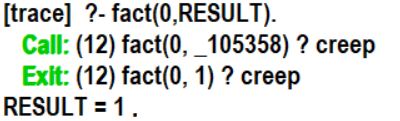
factorial(N1, F1),

F is N \* F1.

**OUTPUT**



**OUTPUT WITH TRACE:**



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(0, 0).

generate\_fib(1, 1).

generate\_fib(N, T) :-

N > 1,

N1 is N - 1,

N2 is N - 2,

generate\_fib(N1, T1),

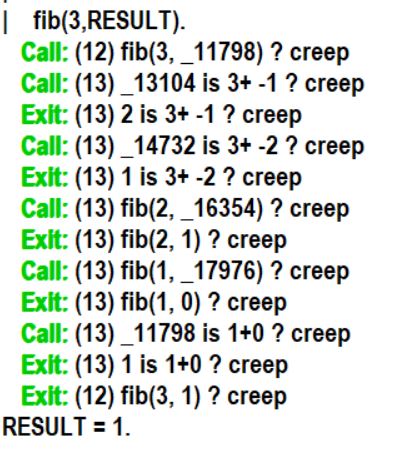
generate\_fib(N2, T2),

T is T1 + T2.

**OUTPUT**



OUTPUT WITH TRACE:



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

**CODE**

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

gcd(X, Y, G) :-

Y > 0,

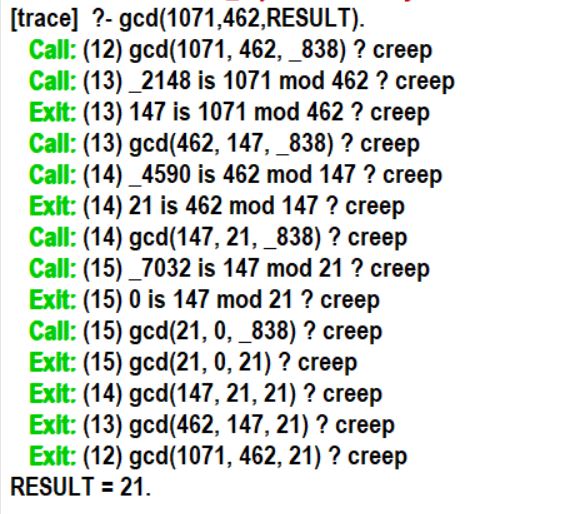
Z is X mod Y,

gcd(Y, Z, G).

**OUTPUT**



OUTPUT WITH TRACE:



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

**CODE**

power(\_, 0, 1).

power(Num, Pow, Ans) :-

Pow > 0,

Pow1 is Pow - 1,

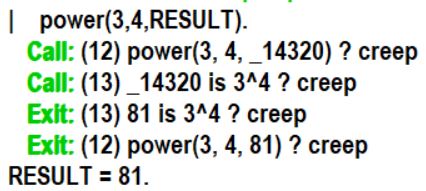
power(Num, Pow1, Ans1),

Ans is Num \* Ans1.

**OUTPUT**

****

OUTPUT WITH TRACE

****

7.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(\_, 0, 0).

multi(N1, N2, R) :-

N2 > 0,

N2\_1 is N2 - 1,

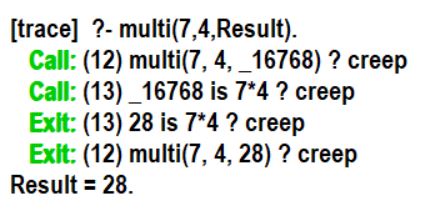
multi(N1, N2\_1, R1),

R is R1 + N1.

**OUTPUT**



**OUTPUT WITH TRACE:**



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|\_]).

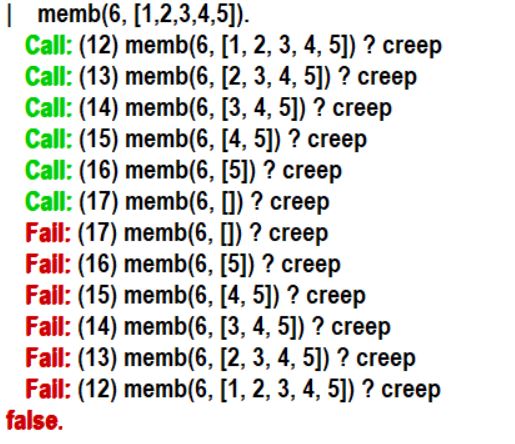
memb(X, [\_|T]) :-

memb(X, T).

**OUTPUT**



OUTPUT WITH TRACE:



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).

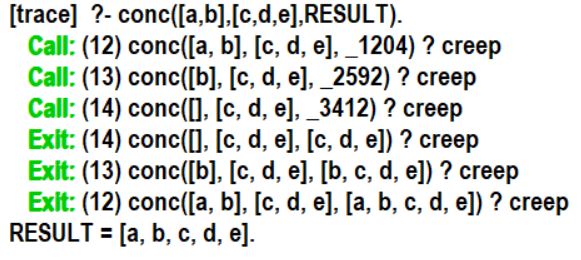
conc([X|L1], L2, [X|L3]) :-

conc(L1, L2, L3).

**OUTPUT**

****

OUTPUT WITH TRACE:



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

R is reversed list.

CODE:

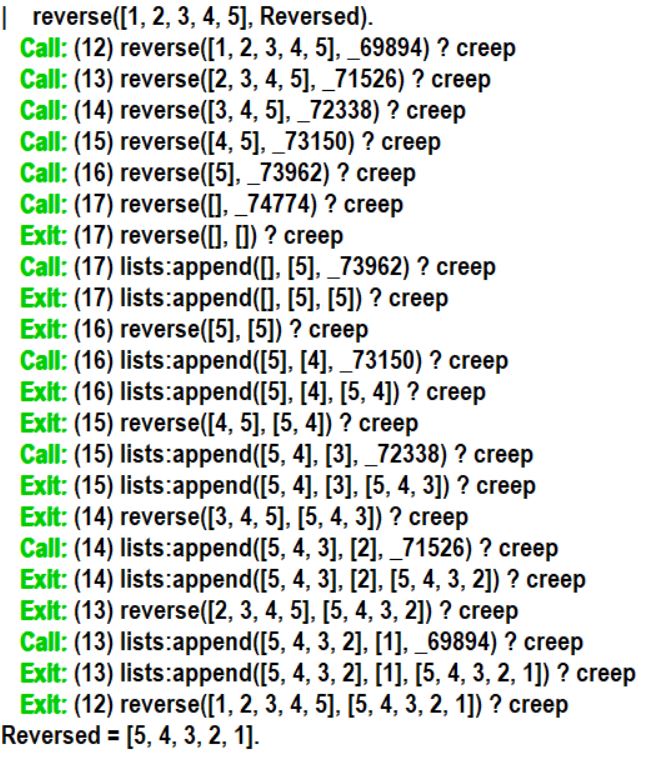
reverse([], []).

reverse([X|Xs], Reversed) :- reverse(Xs, RevXs), append(RevXs, [X], Reversed).

OUTPUT WITHOUT TRACE:



OUTPUT WITH TRACE:



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

CODE:

app([],L,L).

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

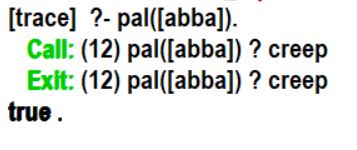
pal([]).

pal([\_]).

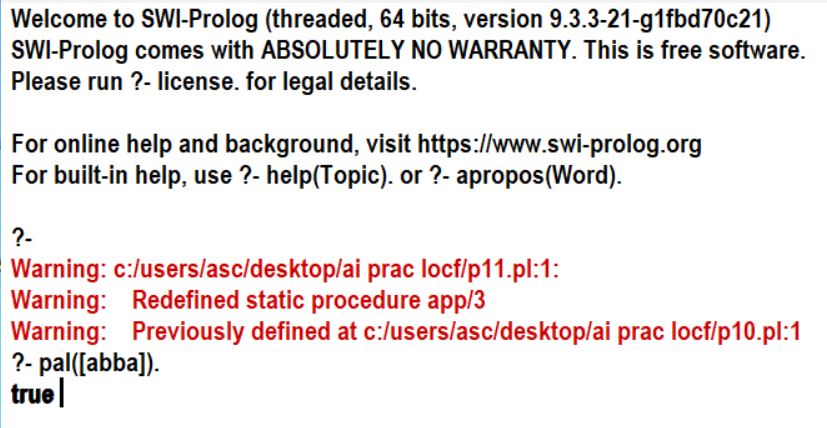
pal(Plist):-app([H|T],[H],Plist),pal(T).

OUTPUT:

OUTPUT WITH TRACE



OUTPUT WITHOUT TRACE



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

CODE:

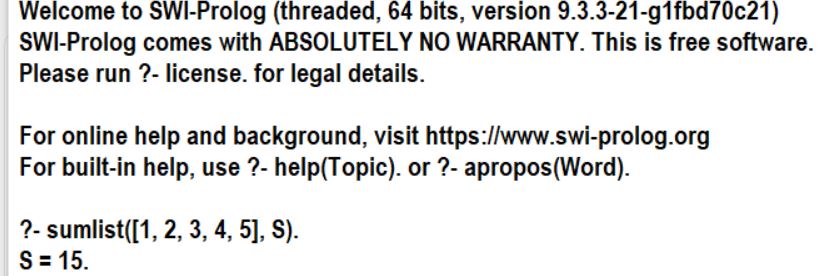
sumlist([], 0).

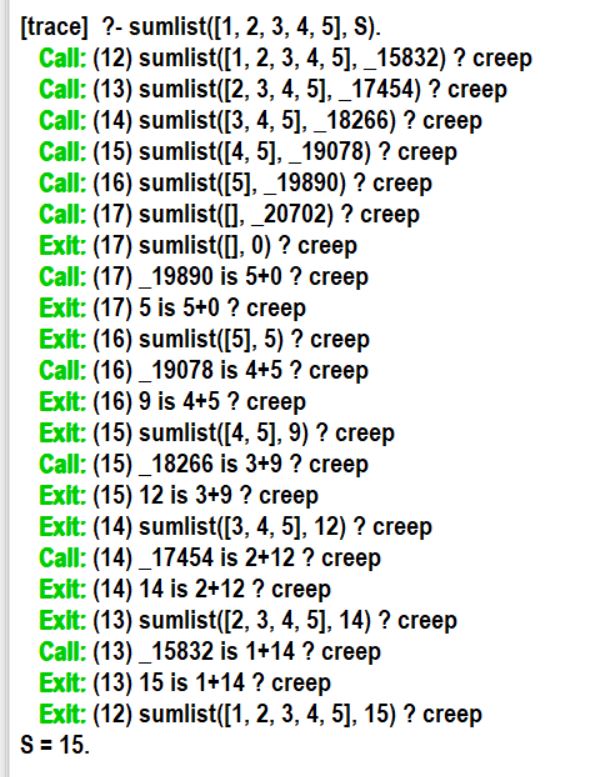
sumlist([Head|Tail], Sum) :-

sumlist(Tail, TailSum),

Sum is Head + TailSum.

OUTPUT WITHOUT TRACE



**OUTPUT WITH TRACE**

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

oddlength([\_]):- !.

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

OUTPUT WITH TRACE:



**OUTPUT WITHOUT TRACE:**



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|\_], H):- !.

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

N > 0,

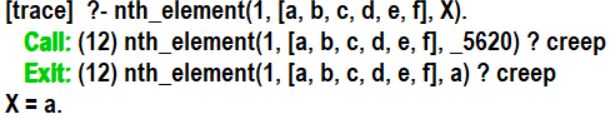
N1 is N - 1,

nth\_element(N1, T, X).

OUTPUT WITHOUT TRACE



OUTPUT WITH TRACE



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

**CODE:**

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

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

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

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

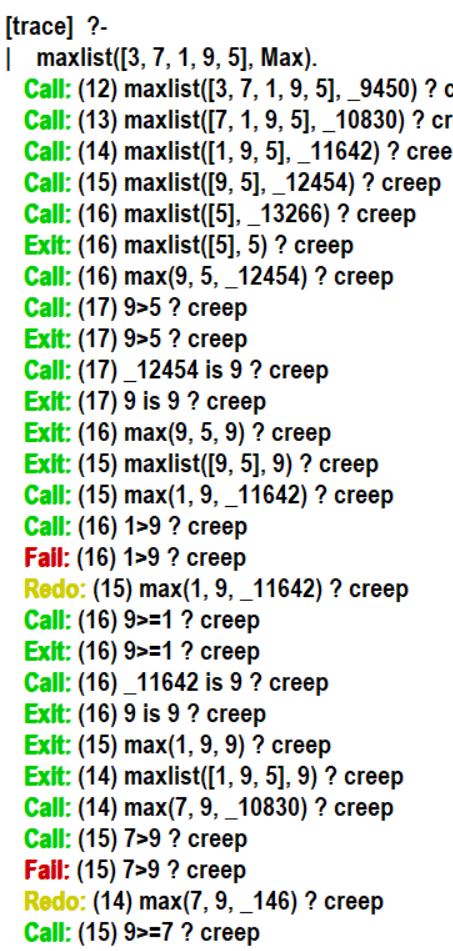
maxlist(T, M1),

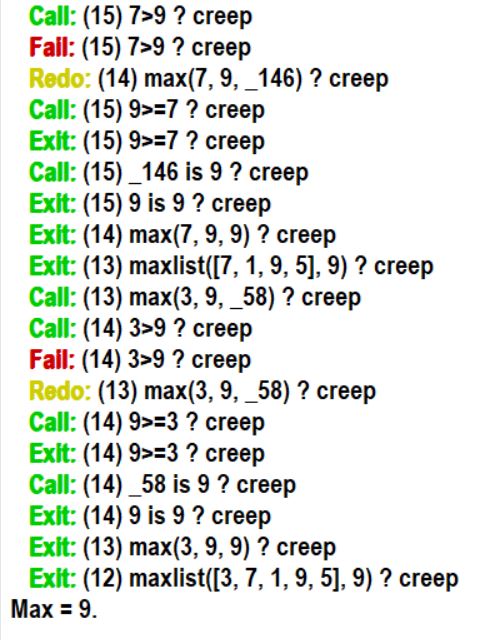
max(H, M1, M).

OUTPUT WITHOUT TRACE:



OUTPUT WITH TRACE





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, N, [Head|Tail], [Head|Result]) :-

N > 1,

N1 is N - 1,

insert\_nth(Item, N1, Tail, Result).

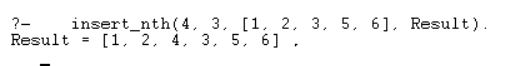
insert\_nth(Item, N, [], Result) :-

N > 0,

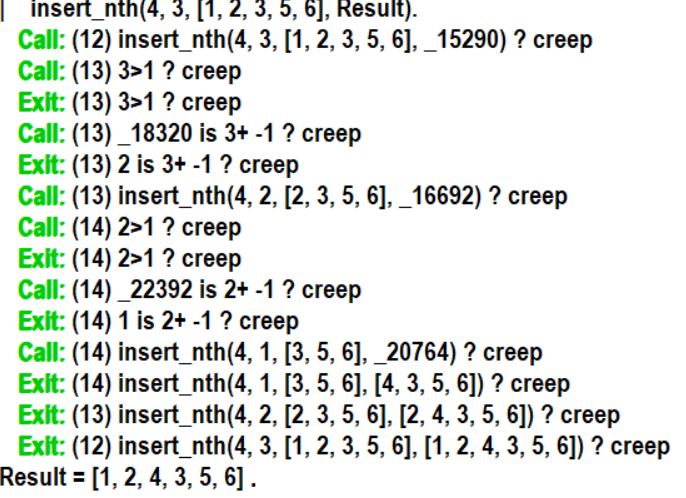
N1 is N - 1,

insert\_nth(Item, N1, [], Result).

**OUTPUT WITHOUT TRACE**



**OUTPUT WITH TRACE**



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, [\_|Tail], Tail).

delete\_nth(N, [Head|Tail], [Head|Result]) :-

N > 1,

N1 is N - 1,

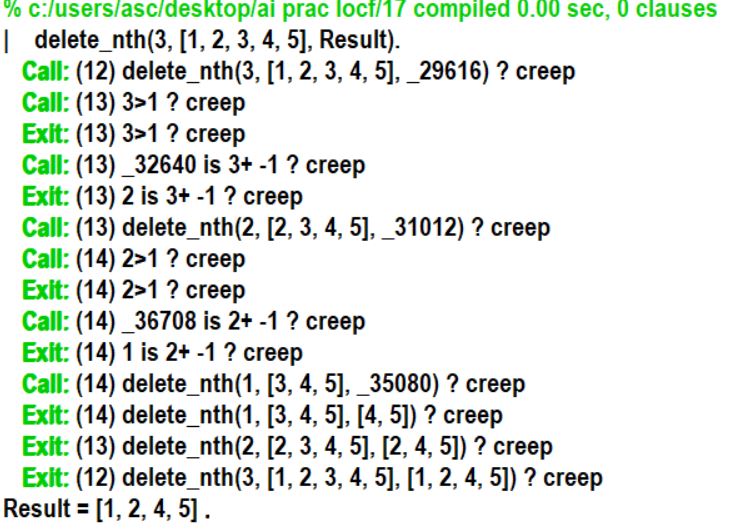
delete\_nth(N1, Tail, Result).

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

**OUTPUT WITHOUT TRACE**



**OUTPUT WITH TRACE**



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:

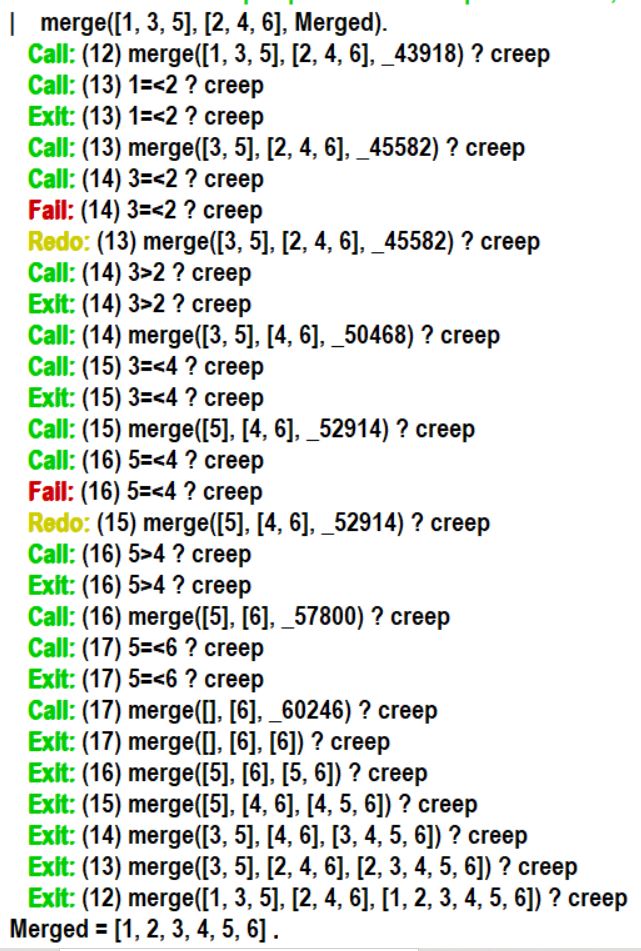
merge([], L, L).

merge(L, [], L) :- L \= [].

merge([X|Xs], [Y|Ys], [X|Zs]) :- X =< Y, merge(Xs, [Y|Ys], Zs).

merge([X|Xs], [Y|Ys], [Y|Zs]) :- X > Y, merge([X|Xs], Ys, Zs).

OUTPUT WIH TRACE:



OUTPUT WITHOUT TRACE:

