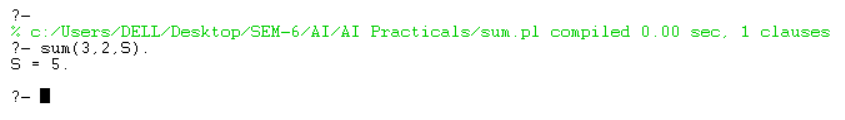
1. Write a prolog program to calculate the sum of two numbers.sum(A,B,S) :- S is A + B.

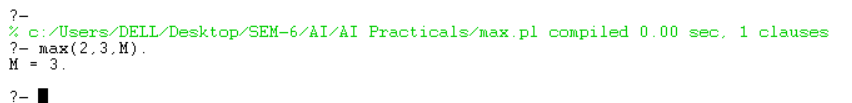
sum(A,B,S) :- S is A + B.



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=X ; M=Y.



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

factorial of a number N.

factorial(0,1).

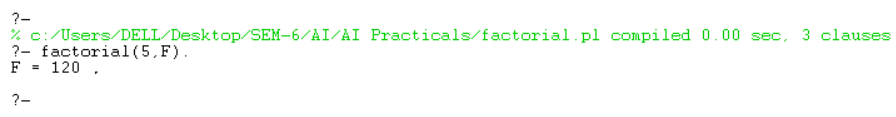
factorial(1,1).

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(0,0).

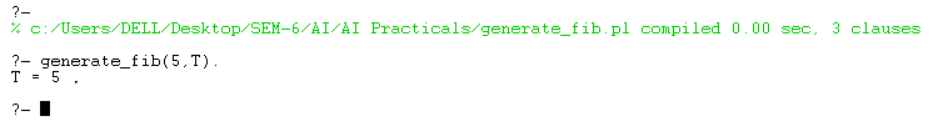
generate\_fib(1,1).

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.

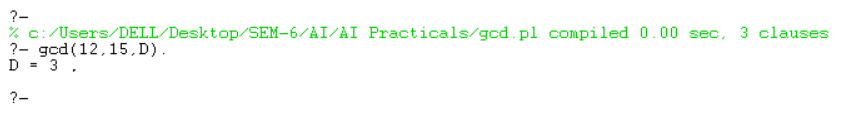


5. Write a Prolog program to implement GCD of two numbers.

gcd(X,X,X).

gcd(X,Y,D) :- X < Y, Y1 is Y - X, gcd(X,Y1,D).

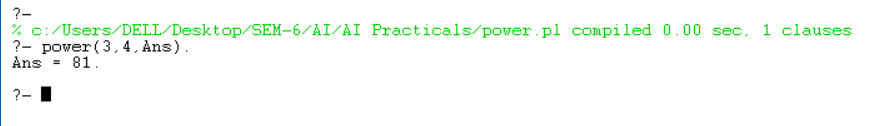
gcd(X,Y,D) :- X > Y, gcd(Y,X,D).



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

to the power Pow to get Ans.

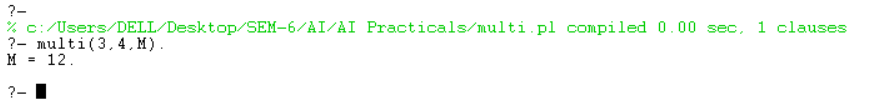
power(Num,Pow,Ans):-Ans is Num\*\*pow.



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(A,B,M):-M is A\*B.

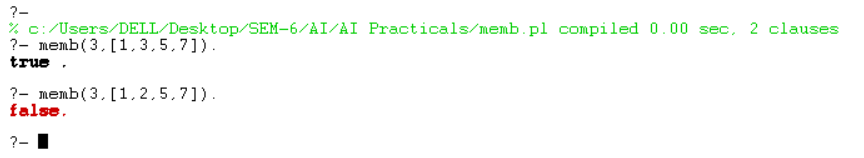


8. Write a Prolog program to implement memb(X, L): to check whether X is a member

of L or not.

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

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

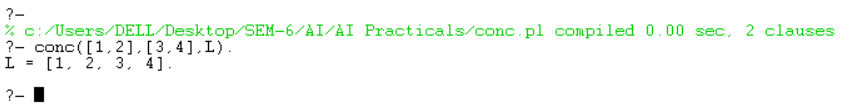


9. Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be

appended with L1 to get the resulted list L3.

conc([],L,L).

conc([H|T],L2,[H|L3]) :- conc(T,L2,L3).



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

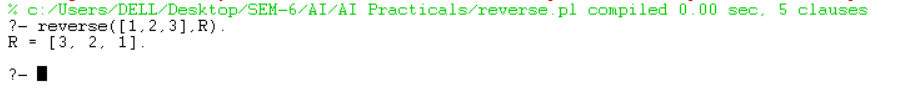
R is reversed list.

conc([],L,L).

conc([H|T],L2,[H|L3]) :- conc(T,L2,L3).

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.

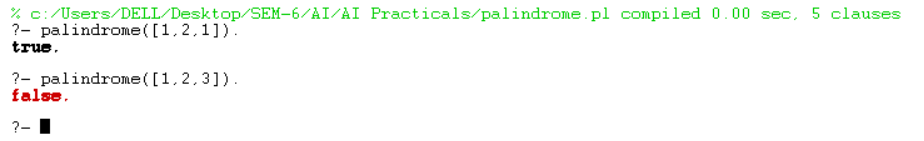
conc([],L,L).

conc([H|T],L2,[H|L3]) :- conc(T,L2,L3).

reverse([],[]).

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

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

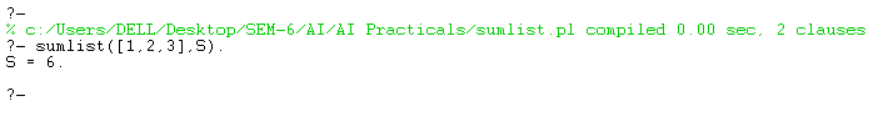


12. Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list

L.

sumlist([], 0).

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



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

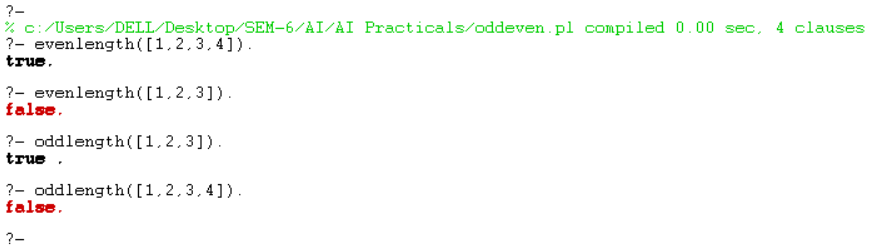
Respectively.

evenlength([]).

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

oddlength([\_]).

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

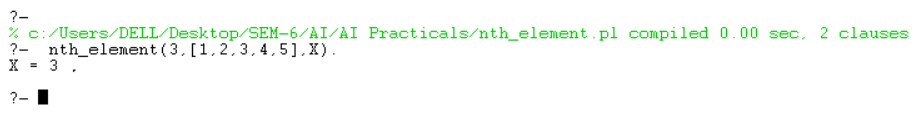


14. Write a Prolog program to implement nth\_element (N, L, X) where N is the desired

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).



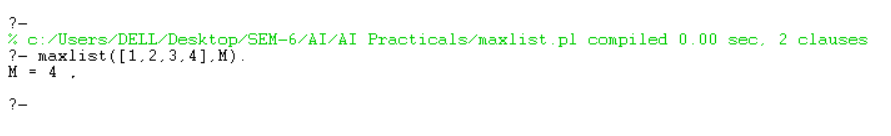
15. Write a Prolog program to implement maxlist(L, M) so that M is the maximum

number in the list.

maxlist([H],H).

maxlist([H|T],R):-maxlist(T,R1),

(H>R1->R=H;R=R1).

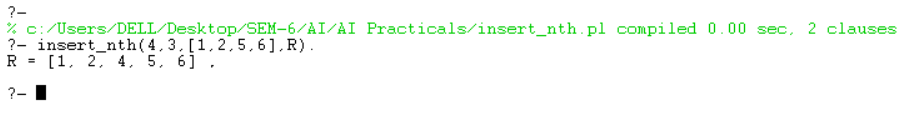


16. Write a prolog program to implement insert\_nth (I, N, L, R) that inserts an item I into

Nth position of list L to generate a list R.

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

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

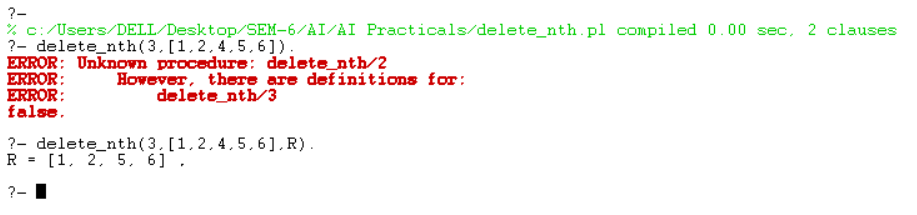


17. Write a Prolog program to implement delete\_nth (N, L, R) that removes the element

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

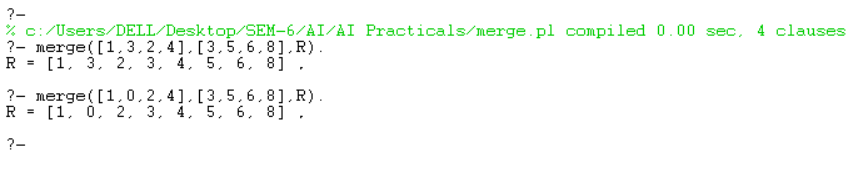
ordered list and L2 is second ordered list and L3 represents the merged list.

merge([], L, L).

merge(L, [], L).

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

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



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

\*\*Illustrate the use of a cut operator.

merge([], L2, L2).

merge(L1, [], L1).

merge([H1|T1], [H2|T2], [H1|T]) :- H1 =< H2, merge(T1, [H2|T2], T), !.

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

% The cut operator is used to prevent backtracking and to ensure that only the first matching % rule is used.

