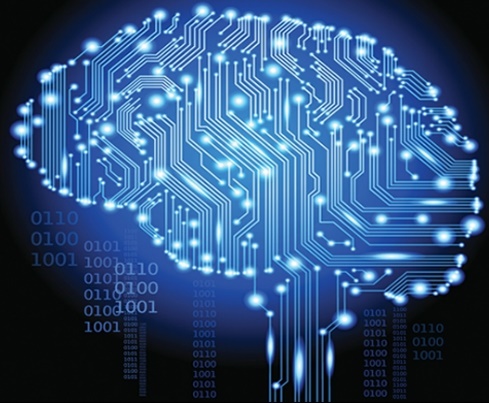
ARTIFICIAL INTELLIGENCE



PRACTICAL FILE

SUBMITTED BY:

**NAME: MADHUBANTI MITRA**

**COLLEGE ROLL NO: 19/78015**

**SEMESTER: VI**

**COURSE: B.SC COMPUTER SCIENCE HONS**

SUBMITTED TO:

**MS. PARUL JAIN,**

**DEPARTMENT OF COMPUTER SCIENCE**

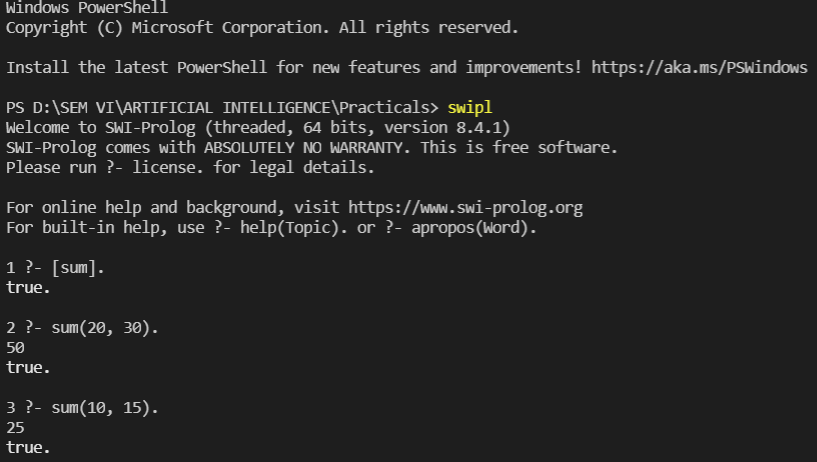
Q1. Write a prolog program to calculate the sum of two numbers.

**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.

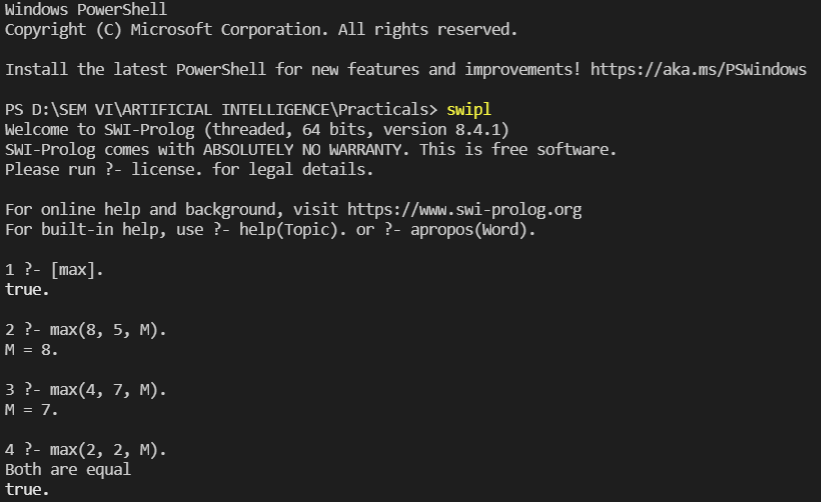
**Code:**

max(X, Y, M) :- X = Y -> write("Both are equal");

X > Y -> M is X;

X < Y -> M is Y.

**Output:**

****

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

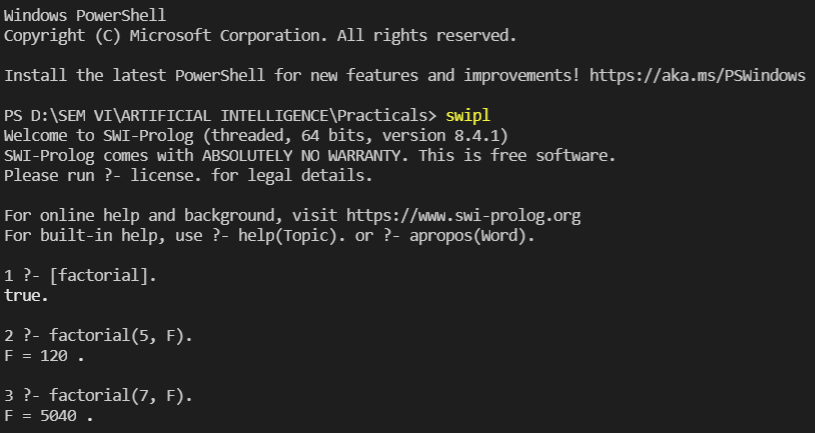
**Code:**

factorial(0, 1).

factorial(N, F) :- N > 0, M is N - 1,

factorial(M, E), F is E \* N.

**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, 1).

generate\_fib(1, 1).

generate\_fib(N, T) :- N1 is N - 1,

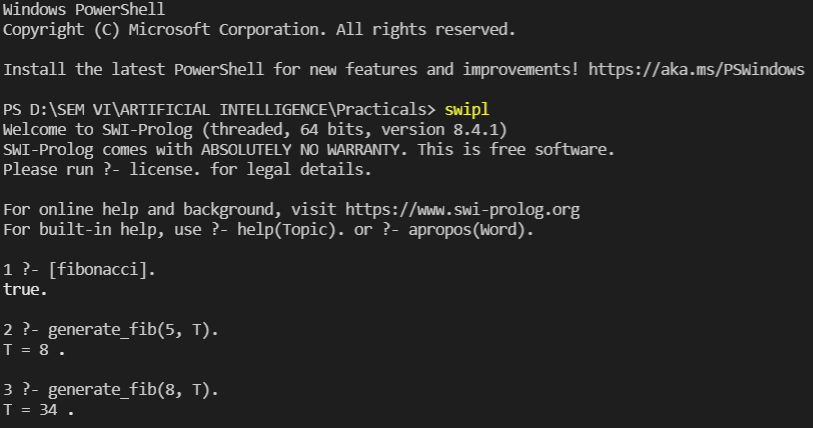
generate\_fib(N1, T1),

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, Y) :- X == 0 -> write(Y);

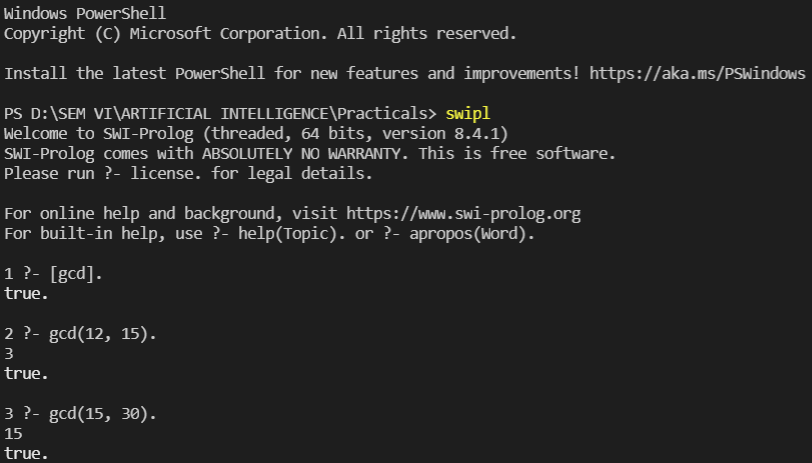
Y == 0 -> write(X);

X == Y -> write(X);

X > Y -> (Z is X - Y, gcd(Z, Y));

Y > X -> (C is Y - X, gcd(X, C)).

**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, Power, 0) :- Power > 0.

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

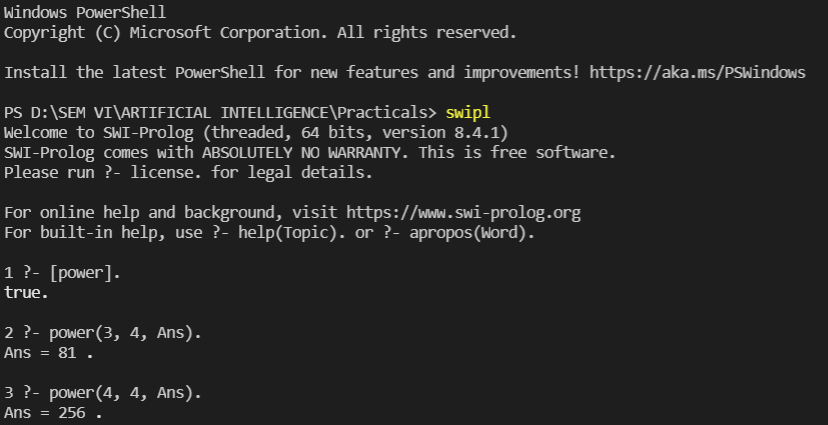
power(Num, Power, Ans) :- Num > 0, Power > 0,

P1 is Power - 1,

power(Num, P1, A1),

Ans is A1 \* 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:**

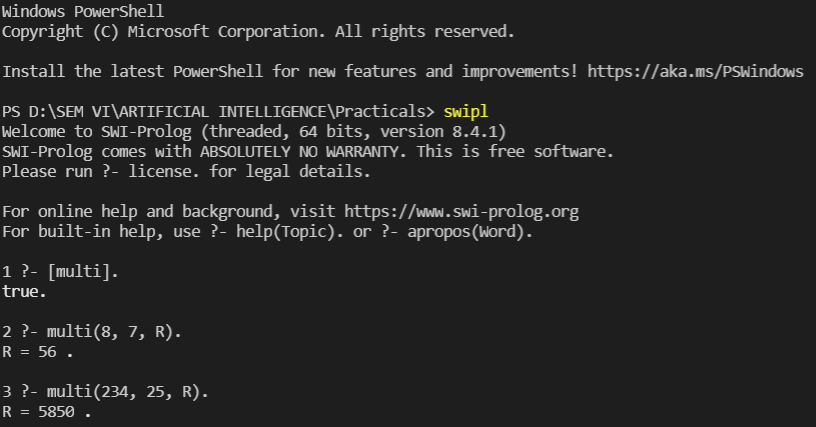
multi(N, 1, N).

multi(N1, N2, R) :- M is N2 - 1,

multi(N1, M, X),

R is X + N1.

**Output:**

****

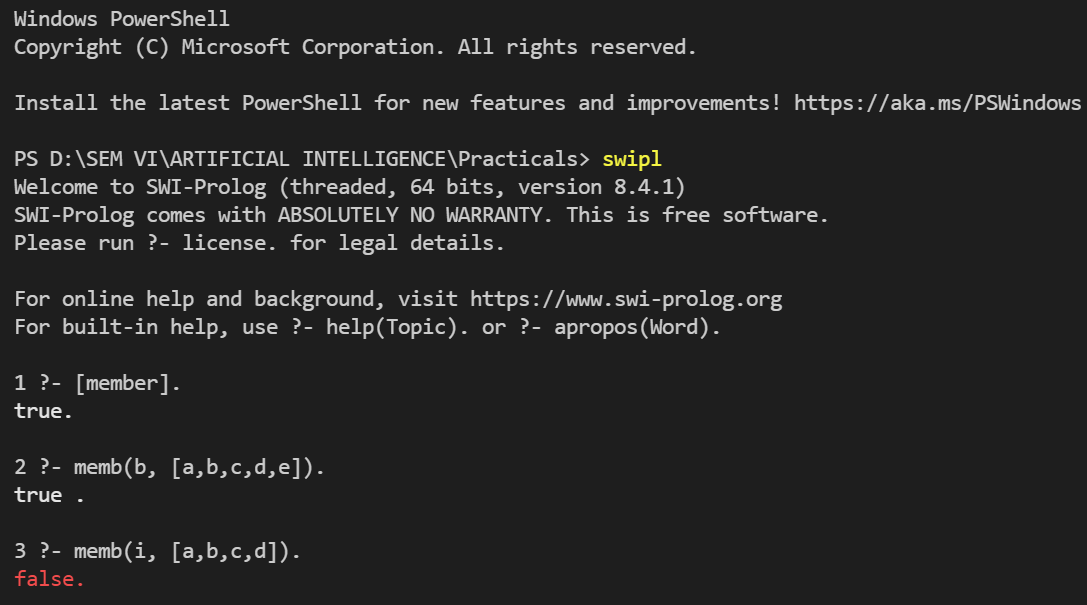
Q8. Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not.

**Code:**

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

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

**Output:**

****

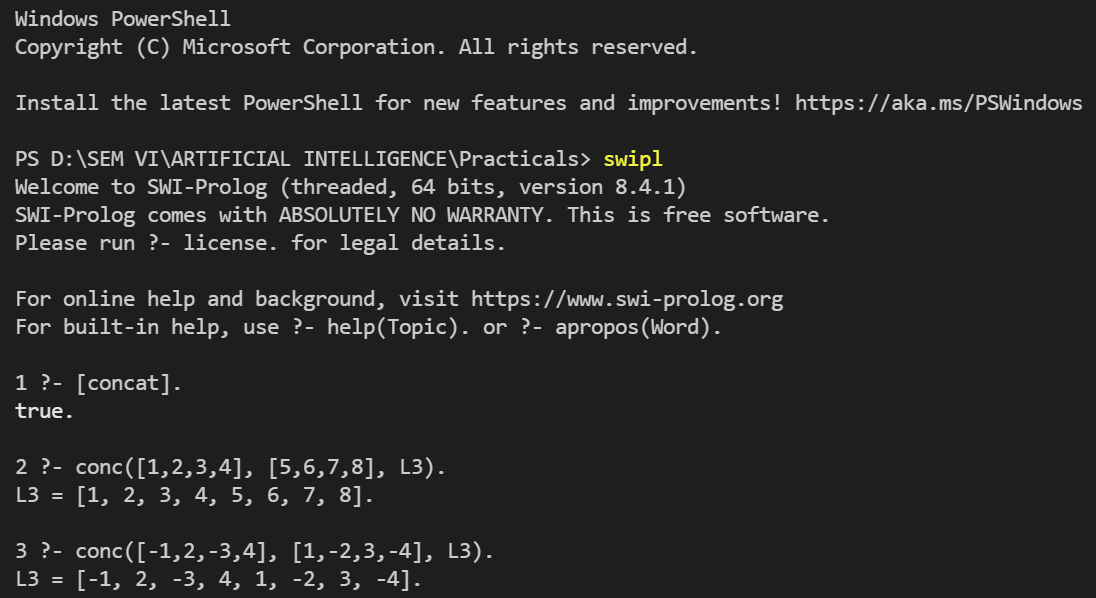
Q9. Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.

**Code:**

conc([], L2, L2).

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

**Output:**

****

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

**Code:**

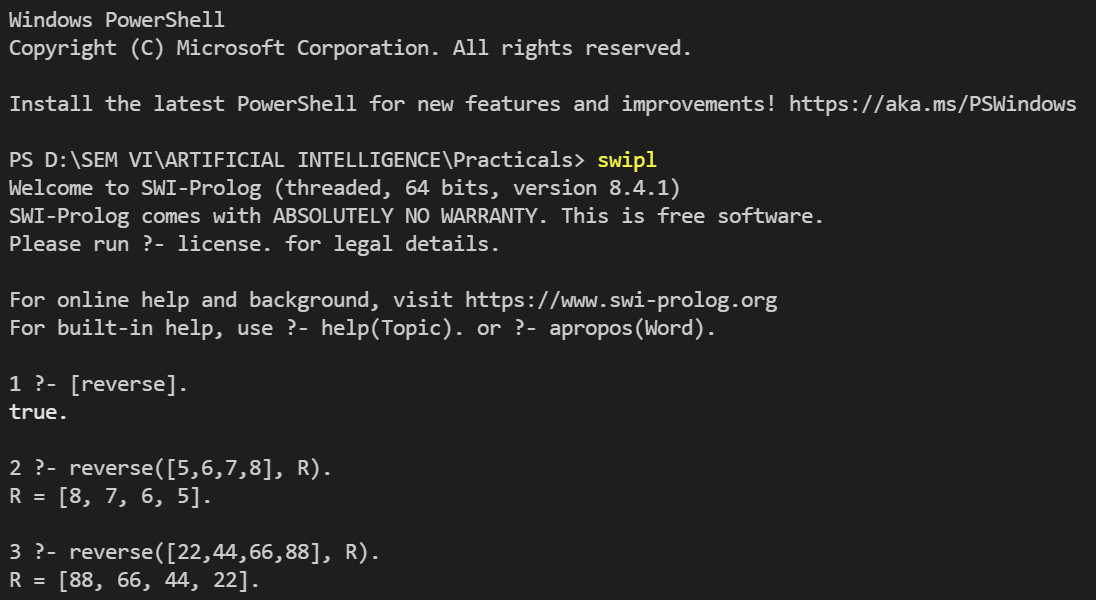
append([],L,L):- !.

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

reverse([],[]):- !.

reverse([H|T],R):- reverse(T,L1),append(L1,[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([]):- write("List is empty.").

palindrome([\_]):- write("List is Palindrome.").

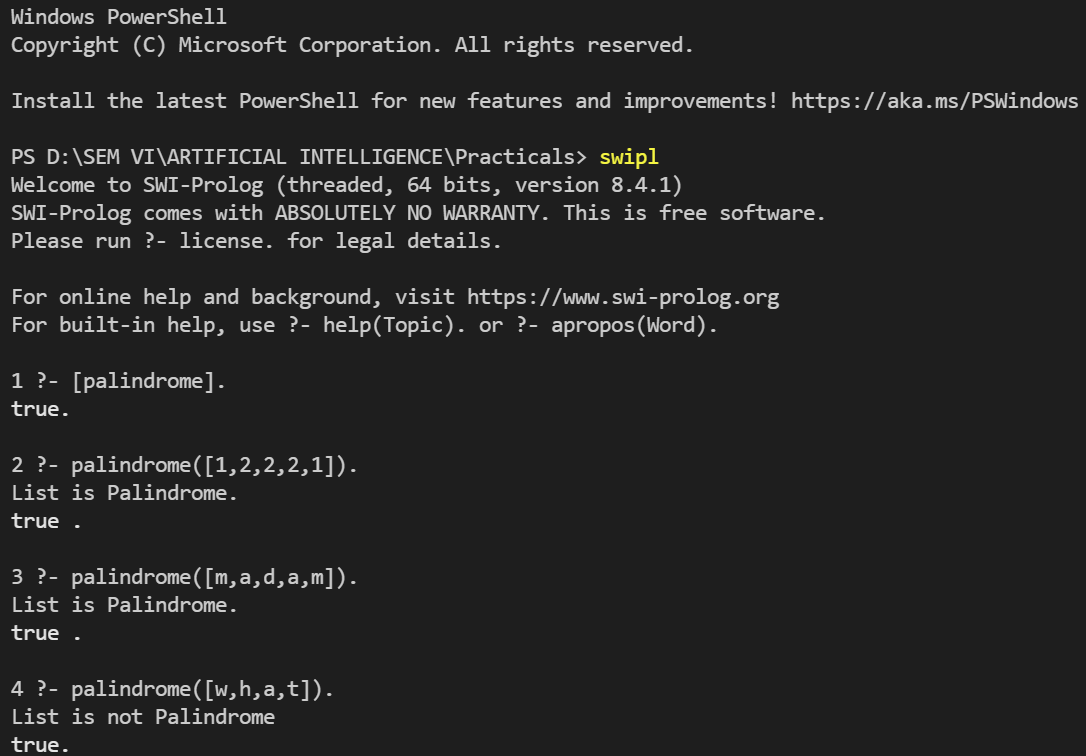
palindrome(L) :- append([H|T], [H], L),

!,

palindrome(T);

write("List is not Palindrome").

**Output:**



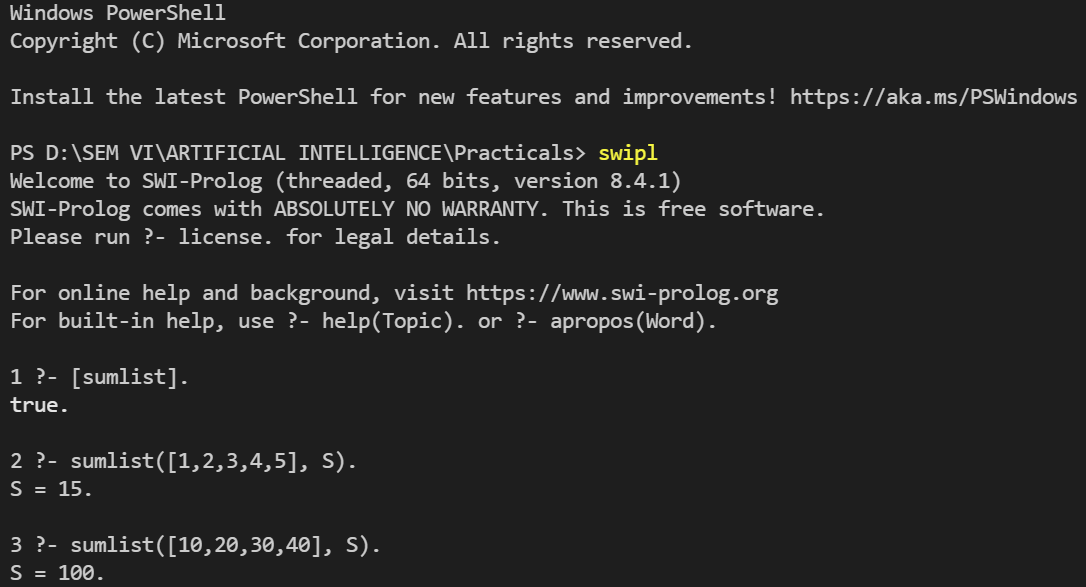
Q12. Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.

**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:**

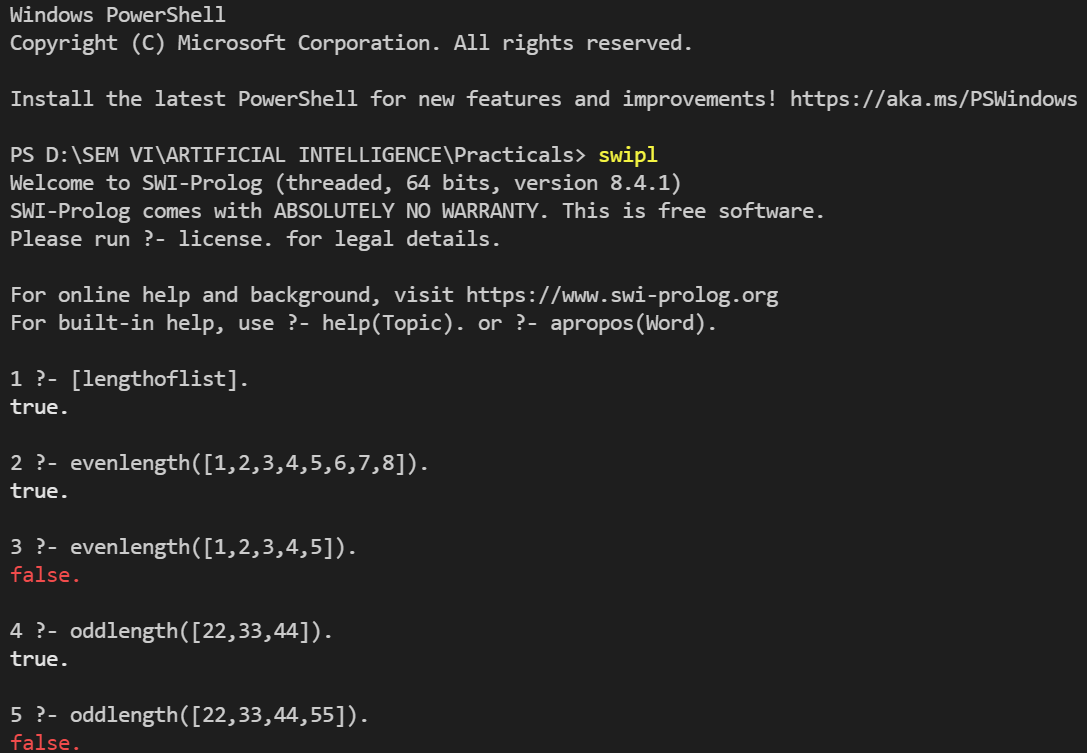
evenlength([]):- !.

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, [\_|L], X):- N1 is N-1,

nth\_element(N1, L, X).

**Output:**

****

Q15. 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:**

****

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.

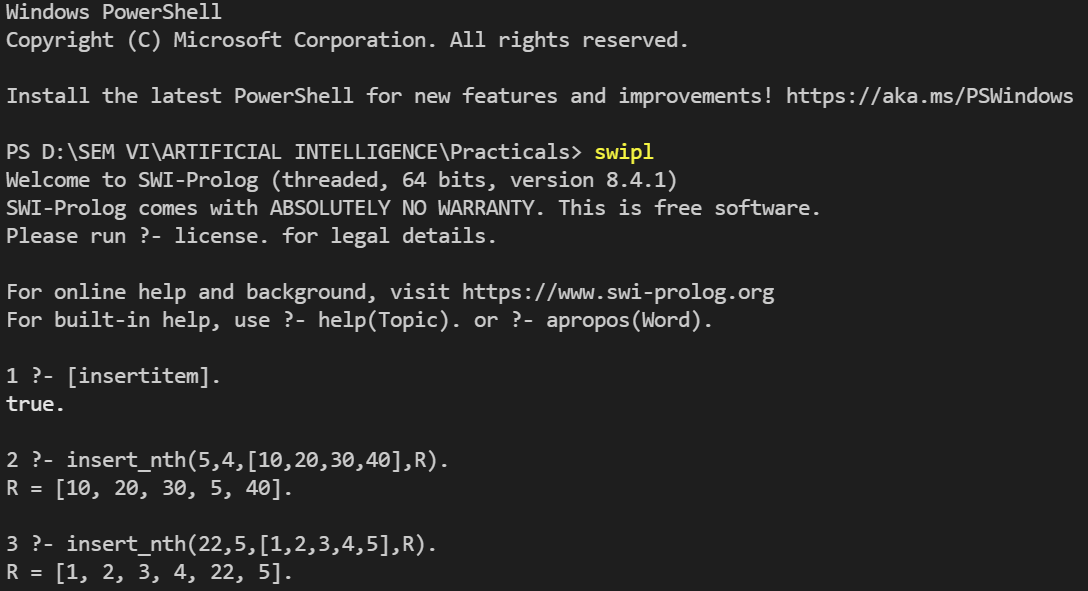
**Code:**

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

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

insert\_nth(I, N1, L, 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.

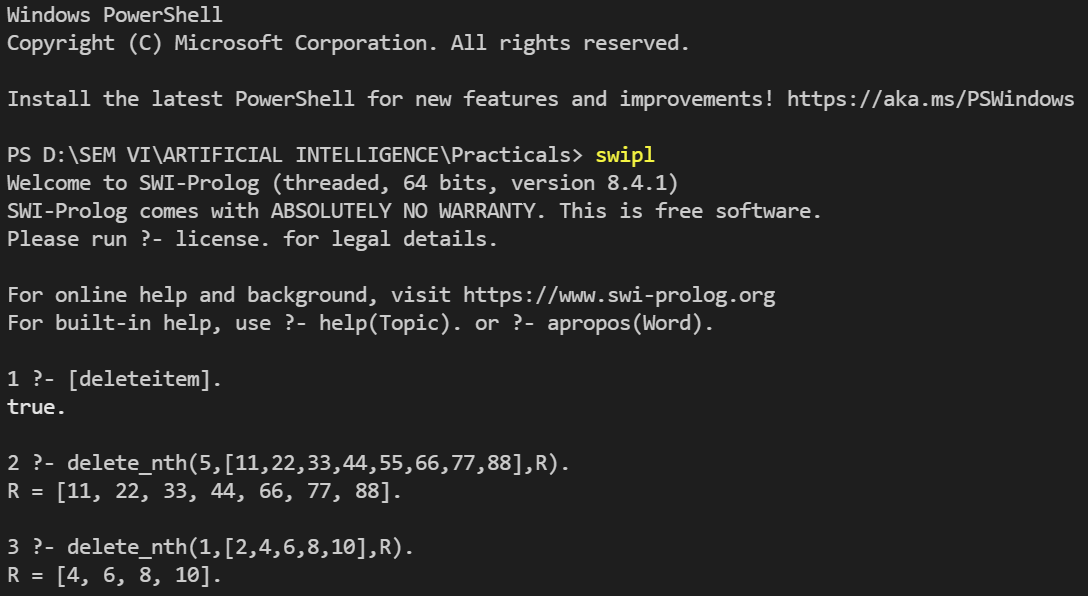
**Code:**

delete\_nth(1, [\_|L], L):- !.

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

delete\_nth(N1, L, 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(X, [], X).

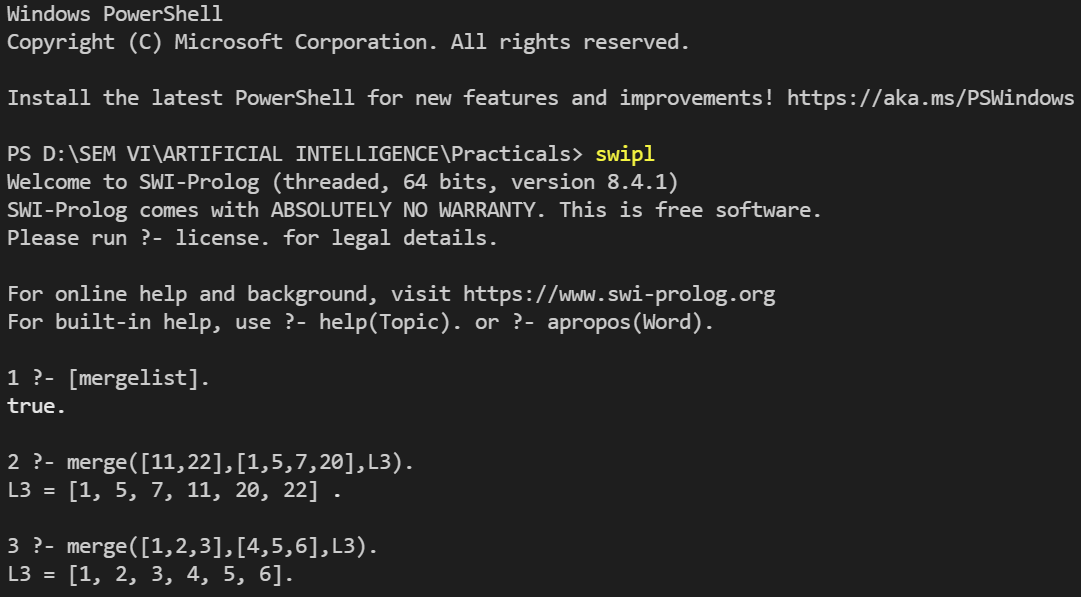
merge([], Y, Y).

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:**

****

**---------- x ----------**