**PRACTICALS**

**ARTIFICIAL INTELLIGENCE**

**NAME: – Pranay Raturi**

**ROLL NUMBER: – 24920**

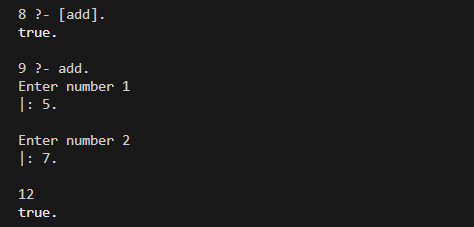
1. Write a prolog program to calculate the sum of two numbers.

CODE:

add:- write('Enter number 1'),nl,read(N1),nl,write('Enter number 2'),nl,read(N2),nl,

Add is N1+N2,write(Add).

OUTPUT:



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

CODE:

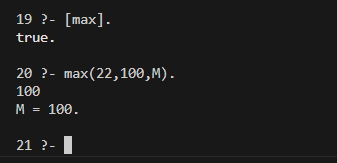
max(X,Y,M):-

X=Y,write('both are equal');

X>Y,M is X,write(M);

X<Y,M is Y,write(M).

OUTPUT:



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

CODE:

fact(0,1).

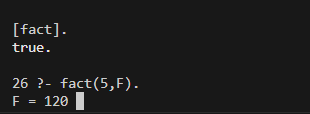
fact(N,F):-

N1 is N-1,

fact(N1, F1),

F is N\*F1.

OUTPUT:



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

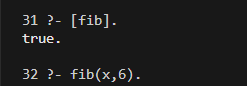
CODE:

fib(0,0).

fib(1,1).

fib(X,N):- N>1,N1 is N-1,N2 is N-2, fib(X1,N1), fib(X2,N2),X is X1+X2.

OUTPUT:



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

CODE:

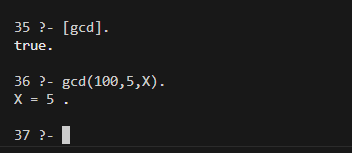
gcd(X,0,X).

gcd(X,Y,Z):-

R is mod(X,Y),

gcd(Y,R,Z).

OUTPUT:



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

CODE:

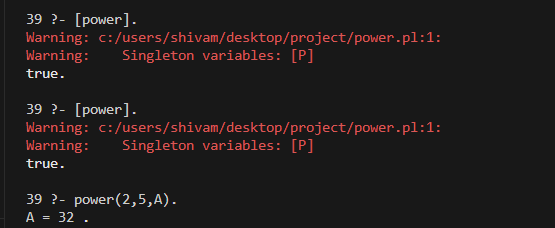
power(0,P,0).

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

power(X,P,A):-

P>0,X>0,P1 is P-1,power(X,P1,Ans),A is Ans\*X.

OUTPUT:



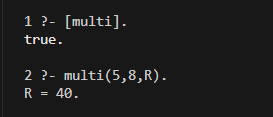
7. Prolog program to implement multi (N1, N2, R) : where N1 and N2 denotes the numbers to be multiplied and R represents the result.

CODE:

multi(N1,N2,R):-

R is N1\*N2.

OUTPUT:



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

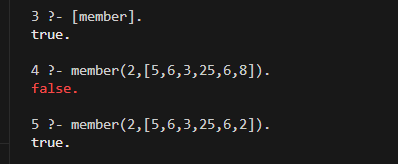
CODE:

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

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

memb(X,B).

OUTPUT:



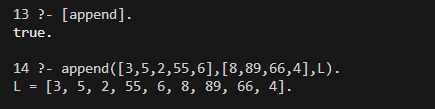
9. Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.

CODE:

append([],L,L).

append([H1|L1],L2,[H1|L3]):-append(L1,L2,L3).

OUTPUT:



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

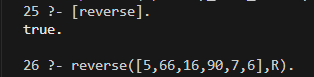
CODE:

reverse([],[]).

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

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

OUTPUT:



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

CODE:

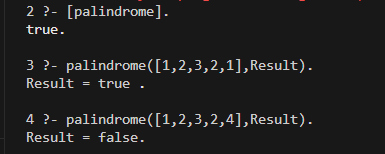
palindrome([],true).

palindrome([\_],true).

palindrome([H|T],Result):- reverse(T,[L|T1]),H = L,palindrome(T1,Result).

palindrome([H|T],false):- reverse(T,[L|\_]),not(H = L).

OUTPUT:



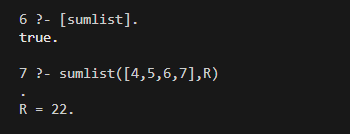
12. Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.

CODE:

sumlist([],0).

sumlist([H|T],R):- sumlist(T,R1), R is R1+H.

OUTPUT:



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:

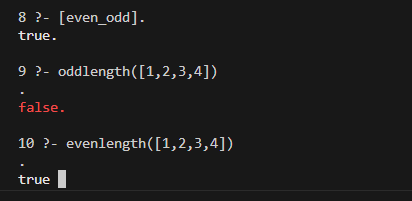
oddlength([\_]).

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

evenlength([]).

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

OUTPUT:



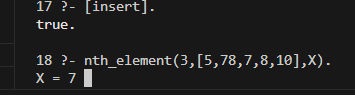
14. Write a Prolog program to implement nth\_element (N, L, X) where N is the desired position, L is a list and X represents the Nth element of L.

CODE:

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

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

OUTPUT:



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

CODE:

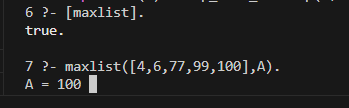
max(A,B,A):- A>B.

max(A,B,B):- B>A.

maxlist([A],A).

maxlist([H,H1|T],R):- maxlist([H1|T],R1), max(H,R1,R).

OUTPUT:



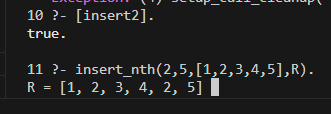
16. Write a prolog program to implement insert\_nth (I, N, L, R) that inserts an item I into Nth position of list L to generate a list R.

CODE:

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

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

OUTPUT:



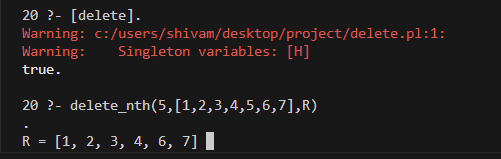
17. Write a Prolog program to implement delete\_nth (N, L, R) that removes the element on Nth position from a list L to generate a list R.

CODE:

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

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

OUTPUT:



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([],[],[]).

merge(L,[],L).

merge([],L,L).

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

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

OUTPUT:

