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

**CODE:**

sum(X,Y,Z):- Z is X+Y.

**OUTPUT:**

C:\Users\welcome\Desktop\DU SEM6\AI\p1op.PNG

**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,M is X.

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

max(X,Y,M):- X=Y,

write("they are equal").

**OUTPUT:**

C:\Users\welcome\Desktop\DU SEM6\AI\p2op.PNG

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

**CODE:**

start :- write('Enter a postiive number:'),read(N1),F is 1,fac(N1,F).

fac(0,F):-write('Factorial is'),write(F).

fac(N,F):-N\=0,

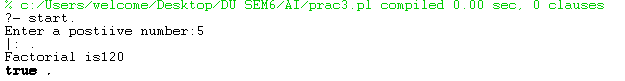
    NewF is F\*N,

    NewN is N-1,

    fac(NewN,NewF).

fac(1,F):-write('Factorial is',F).

**OUTPUT :**



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

**CODE:**

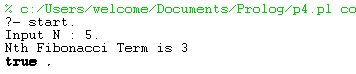
start1:-write('Enter N : '),read(N), fibo(N,T), write('Term is '),write(T). fibo(0,0). fibo(1,1).

fibo(N,T):-N>1,

N1 is N-1, fibo(N1,R1), N2 is N-2, fibo(N2,R2),

T is R1+R2.

**OUTPUT:**



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

**CODE:**

gcdcalc:-

        write("Enter x : "),    read(X),

        write("Enter y : "),    read(Y),

        gcd(X,Y).

gcd(X,X):-      write("Result = "), write(X).

gcd(0,\_):-      write("Result = 0").

gcd(\_,0):-      write("Result = 0").

gcd(X,Y):-

                X>=Y,

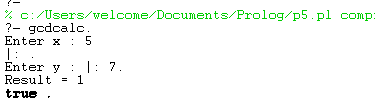
                Xn is X-Y,

                gcd(Xn,Y);

                Xn is Y-X,

                gcd(X,Xn).

**OUTPUT :**



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

**CODE:**

powcalc:-

        write("Enter number : "),       read(Num),

        write("Enter power : "),        read(Pow),

        Ans is 1,

        pwr(Num,Pow,Ans).

pwr(\_,0,Ans):- write(Ans).

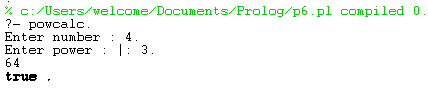
pwr(Num,Pow,Ans):-

                Pow2 is Pow-1,

                NewAns is Ans\*Num,

                pwr(Num,Pow2,NewAns).

**OUTPUT:**



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

**CODE:**

multiply:-

    write("Enter number N1: "),

    read(N1),

    nl,

    write("Enter number N2: "),

    read(N2),

    nl,

    multi(N1,N2).

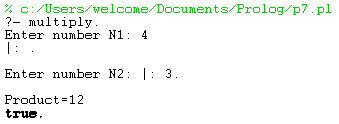
multi(N1,N2):-

    R is N1 \* N2,

    write("Product="),

    write(R).

**OUTPUT:**



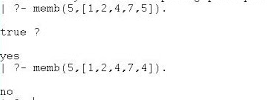
1. **Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not.**

**CODE:**

memb(X,[X,Tail]).

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

**OUTPUT:**



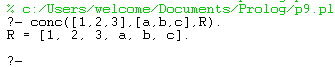
1. **Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.**

**CODE:**

conc([], List, List).

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

**OUTPUT:**



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

**CODE:**

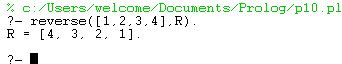
append([],L,L).

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

reverse([],[]).

reverse([H|T],R):-reverse(T,L1),append(L1,[H],R).

**OUTPUT:**



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

**CODE:**

append([],L,L).

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

pali([]).

pali([\_]).

pali(Plist):-append([H|T],[H],Plist),

            pali(T).

**OUTPUT:**

C:\Users\welcome\Desktop\DU SEM6\AI\p11op.PNG

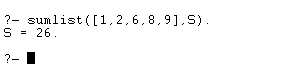
1. **Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.**

**CODE:**

sumlist([],0).

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

**OUTPUT:**



1. **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:**

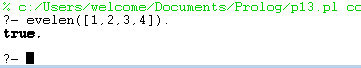
evelen([]).

evelen([\_|[\_|List]]):- evelen(List).

oddlen([\_]).

oddlen([\_|[\_|List]]):- oddlen(List).

**OUTPUT:**



1. **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:**

start:-

    write("Enter the list : "),

    read(L),

    write("Enter N :"),

    read(N),

    nth\_element(N,X,L),

    write("Nth element is : " ),write(X).

nth\_element(0,X,[X|\_]).

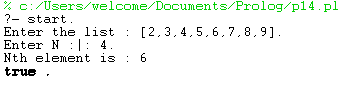
nth\_element(N,X,[\_|Xs]):-

    N>0,

    N1 is N-1,

    nth\_element(N1,X,Xs).

**OUTPUT:**



1. **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:**

C:\Users\welcome\Desktop\DU SEM6\AI\p15op.PNG

1. **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,List,[I,List]).

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

                             insert\_nth(I,N1,T,R).

**OUTPUT:**

C:\Users\welcome\Desktop\DU SEM6\AI\p16op.PNG

1. **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:**

delte(1,[\_|T],T).

delte(P,[X|Y],[X|R]):-

 P1 is P-1,

 delte(P1,Y,R).

**OUTPUT:**

C:\Users\welcome\Desktop\DU SEM6\AI\p17op.PNG

1. **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:**

