Artificial Intelligence (CS308)

Assignment - 4

**U19CS012**

1.) W.A.P.P to Find **Factorial** of a Number. {**W.A.P.P** – Write a Prolog Program}

**Prolog Code**

*% W.A.P.P to Find Factorial of a Number. [U19CS012]*

main *:-*

   write("Enter a Positive Integer : "),

   read(N),

   fact(N,Ans),

   write("Factorial of "),

   write(N),

   write(" : "),

   write(Ans),

   nl*.*

*% 0! = 1*

fact(0, Ans) *:-*

   Ans is 1*.*

*% n! = n\*(n-1)!...0!*

fact(N, Ans) *:-*

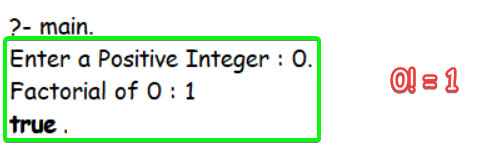
   N>0,

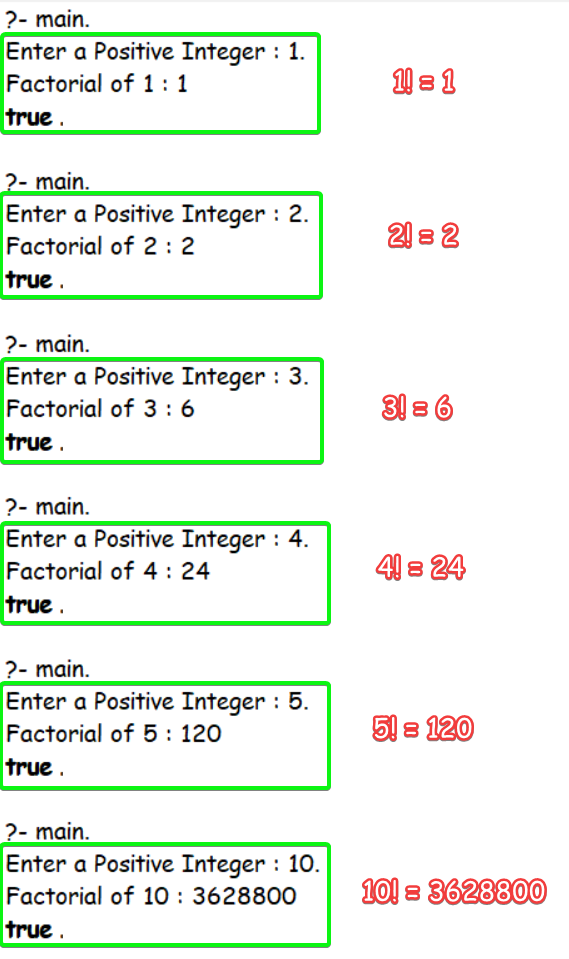
   New\_N is N-1,

   fact(New\_N,X1),

   Ans is X1\*N*.*

**Output**





2.) W.A.P.P to Print **Fibonacci** **Series**.

The Fibonacci sequence f (1), f (2), f (3)…is: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55....

?- fib (6, R).

R = 8

**Prolog Code**

*% W.A.P.P to Print Fibonacci Series. [U19CS012]*

main*:-*

        write("Enter 'n' for nth Fibonacci Term (n>0) : "),

        read(N),

        fib(N,X),

        write("Fibonacci Series "),

        write(N),

        write(" th Term"),

        write(" is : "),

        write(X),

        nl*.*

*% fib(1) = 1*

fib(1,Ans) *:-*

        Ans is 1*.*

*% fib(2) = 1*

fib(2,Ans) *:-*

        Ans is 1*.*

*% fib(n) = fib(n-1) + fib(n-2) ... if(n>2)*

fib(N,Ans) *:-*

        N>2,

        Y is N-1,

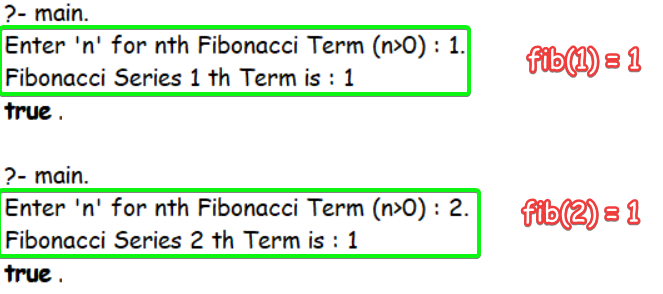
        Z is N-2,

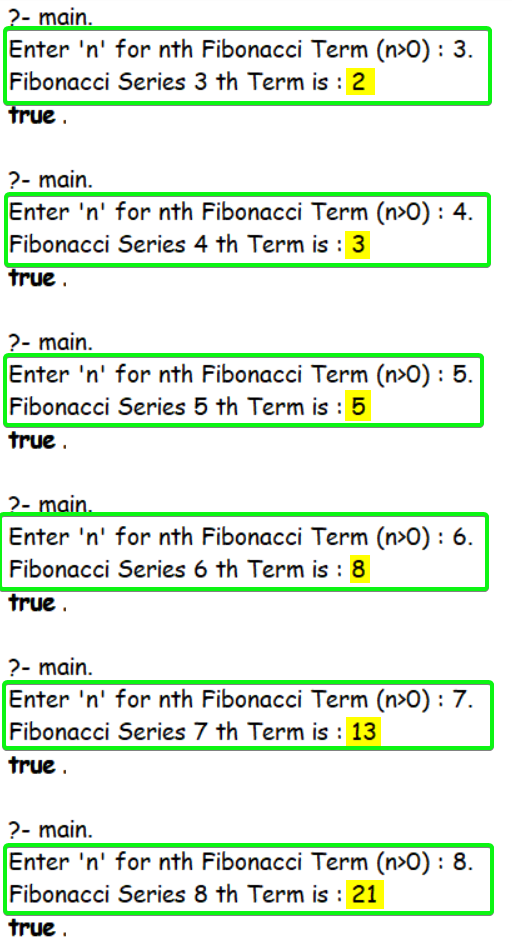
        fib(Y,X1),

        fib(Z,X2),

        Ans is X1+X2*.*

**Output**





3.) W.A.P.P to finding the **Greatest Common Divider** (GCD) and **Least Common Multiple** (LCM) of two integers.

**Prolog Code**

*% W.A.P.P to finding the GCD and LCM of two integers. [U19CS012]*

*% gcd(N,M) \* lcm(N,M) = N\*M  ... (1)*

main *:-*

    write("Calculate GCD & LCM of Two Numbers!"),

    nl,

    write("Enter Number 1 : "),

    read(N),

    nl,

    write("Enter Number 2 : "),

    read(M),

    nl,

    gcd(N,M,X),

    write("GCD Of "),

    write(N),

    write(" & "),

    write(M),

    write(" is : "),

    write(X),nl,

    Z is N\*M,

    Y is Z/X,

    write("LCM of "),

    write(N),

    write(" & "),

    write(M),

    write(" is : "),

    write(Y),

    nl*.*

*% int gcd(int n, int m)*

*%    return m == 0 ? n : gcd(m, n % m);   ...(2)*

*% Base Case when M = 0*

gcd(N, 0, Ans) *:-*

    Ans is N*.*

*% gcd(n,m) = gcd(m, n%m)*

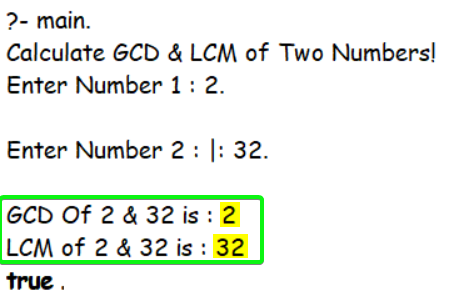
gcd(N, M, Ans) *:-*

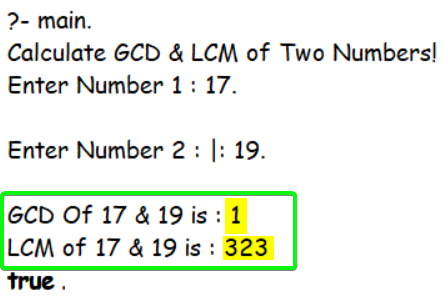
    M>0,

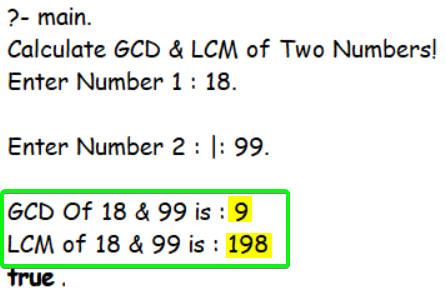
    Y is mod(N, M),

    gcd(M, Y, Ans)*.*

**Output**







4. W.A.P.P.

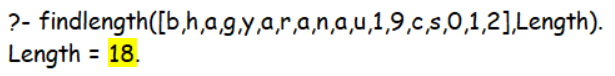
1. To find length of the list.

findlength([],0)*.*

findlength([*\_*|T], N) *:-*

    findlength(T,X),

    N is X+1*.*

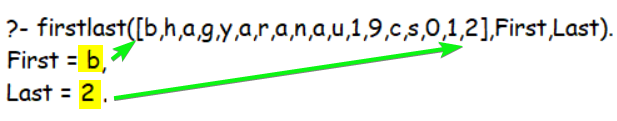


1. To find first and last element of the list.

firstlast([],[],[])*.*

firstlast([H],H,H)*.*

firstlast([H|T],H,L) *:-* firstlast(T,*\_*,L)*.*



1. To find the nth element of the list.

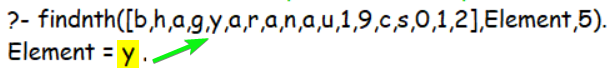
findnth([H|*\_*],H,1)*.*

findnth([*\_*|T],X,N) *:-*

    N1 is N-1,

    N1 > 0,

    findnth(T,X,N1)*.*



1. To increment each number in the list.

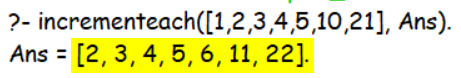
*% increment each element X is input Y is output*

incrementeach([],[])*.*

incrementeach([X|Xs],[Y|Ys]) *:-*

    (number(X) *->* Y is X+1),

    incrementeach(Xs,Ys)*.*



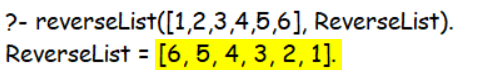
1. To reverse the list.

*% reverse the list*

reverseList(Inputlist,Outputlist) *:-* reverse(Inputlist,[],Outputlist)*.*

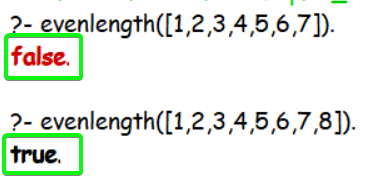
reverse([],Outputlist,Outputlist)*.*

reverse([Head|Tail],List1,List2) *:-* reverse(Tail,[Head|List1],List2)*.*



1. To verify if a list has an even number of elements.

evenlength([H|T]) *:-* findlength([H|T],X), 0 is mod(X,2)*.*



1. To count vowels in the list.

*% find number of vowels*

vowel(a)*.*

vowel(e)*.*

vowel(i)*.*

vowel(o)*.*

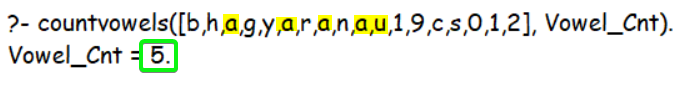
vowel(u)*.*

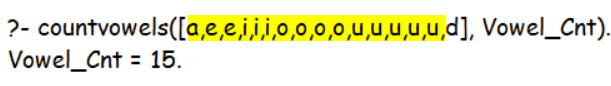
countvowels([],0)*.*

*% Exclamation point ! denotes Cut in Prolog*

*% a special goal that always succeeds, and blocks backtracking for all branches above it that may have alternatives.*

countvowels([H|T],X) *:-* (countvowels(T,Y),vowel(H),X is Y+1,*!*);(countvowels(T,X))*.*





1. To remove duplicates from the list.

*% remove duplicates*

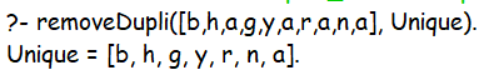
chk(X,[H|T]) *:-* ((X=H,*!*);chk(X,T))*.*

removeDupli([],[])*.*

removeDupli([H|T],X) *:-*

    ((removeDupli(T,Y),not(chk(H,T)),append([H],Y,X),*!*);

    (removeDupli(T,X)))*.*



**SUBMITTED BY**: U19CS012

BHAGYA VINOD RANA