PROLOG

PRACTICALS

PRACTICAL 1

/\*sum(X,Y,R):- R is X+Y.

\*/

sum :-

write('Enter 1st number : '),

read(X),

write('Enter 2nd number : '),

read(Y),

Z is X+Y,

write('Sum of '),write(X),write(' and '),write(Y),write(' is '),write(Z),nl,

write('Do you want to add more numbers (yes/no) : '),

read(Ch),

cont(Ch).

cont(yes) :- sum.

cont(no) :- !.

cont(Ch) :- write('Sorry, you entered wrong choice !'),!.

PRACTICAL 2

/\* max(X,Y,M):- M is X, X>=Y.

max(X,Y,M):- M is Y, X<Y. \*/

input :-

write('Enter 1st number : '),

read(X),

write('Enter 2nd number : '),

read(Y),

max(X,Y,M).

max(X,Y,M):-

M is X, X>=Y,

output(X,Y,M).

max(X,Y,M):-

M is Y, X<Y,

output(X,Y,M).

output(X,Y,M) :-

write('Maximum of '),write(X),write(' and '),write(Y),write(' is '),write(M),nl,

write('Do you want to continue ? (yes/no) : '),

read(Ch),

cont(Ch).

cont(yes) :- input.

cont(no) :- !.

cont(Ch) :- write('Sorry, you entered wrong choice !'),!.

PRACTICAL 3

/\*factorial(0,1).

factorial(N,R):-N>0,N1 is N-1,factorial(N1,Res),R is N\*Res.

\*/

input :-

write('Enter the number : '),

read(N),

factorial(N,R),

output(N,R).

factorial(0,1).

factorial(N,R):-N>0,N1 is N-1,factorial(N1,Res),R is N\*Res.

output(N,R) :-

write('Factorial of '),write(N),write(' is '),write(R),nl,

write('Do you want to continue ? (yes/no) : '),

read(Ch),

cont(Ch).

cont(yes) :- input.

cont(no) :- !.

cont(Ch) :- write('Sorry, you entered wrong choice !'),!.

PRACTICAL 4

/\*generate\_fib(1,0).

generate\_fib(2,1).

generate\_fib(N,T):-N>2,N1 is N-1,N2 is N-2,generate\_fib(N1,T1),generate\_fib(N2,T2),T is T1+T2.

\*/

input :-

write('Enter the value of N (Nth term) : '),

read(N),

generate\_fib(N,T),

output(N,T).

generate\_fib(1,0).

generate\_fib(2,1).

generate\_fib(N,T):-N>2,N1 is N-1,N2 is N-2,generate\_fib(N1,T1),generate\_fib(N2,T2),T is T1+T2.

output(N,T) :-

nl,write(N),write('th term of fibonacci series : '),write(T),nl,nl,

write('Do you want to continue ? (yes/no) : '),

read(Ch),

cont(Ch).

cont(yes) :- input.

cont(no) :- !.

cont(Ch) :- write('Sorry, you entered wrong choice !'),!.

PRACTICAL 5

/\*

gcd(0, X, X) :-!.

gcd(X, 0, X) :-!.

gcd(X, X, X) :-!.

gcd(M, N, X) :- N>M, Y is N-M, gcd(M, Y, X).

gcd(M, N, X) :- N<M, Y is M-N, gcd(Y, N, X).

gcd(X,0,X).

gcd(X,Y,G) :-

R is mod(X,Y),

gcd(Y,R,G).

\*/

input :-

write('Enter 1st number : '),

read(X),

write('Enter 2nd number : '),

read(Y),

gcd(X,Y,G),

output(X,Y,G).

gcd(0, X, X) :-!.

gcd(X, 0, X) :-!.

gcd(X, X, X) :-!.

gcd(M, N, X) :- N>M, Y is N-M, gcd(M, Y, X).

gcd(M, N, X) :- N<M, Y is M-N, gcd(Y, N, X).

/\*gcd(X,0,X).

gcd(X,Y,G) :-

R is mod(X,Y),

gcd(Y,R,G).

\*/

output(X,Y,G) :-

write('GCD of '),write(X),write(' and '),write(Y),write(' is '),write(G),nl,

write('Do you want to continue ? (yes/no) : '),

read(Ch),

cont(Ch).

cont(yes) :- input.

cont(no) :- !.

cont(Ch) :- write('Sorry, you entered wrong choice !'),!.

PRACTICAL 6

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

power :-

write('Enter Number : '),

read(X),

write('Enter Power : '),

read(Y),

Z is X\*\*Y,

write(X),write(' to the power '),write(Y),write(' is '),write(Z),nl,

write('Do you want to continue ? (yes/no) : '),

read(Ch),

cont(Ch).

cont(yes) :- power.

cont(no) :- !.

cont(Ch) :- write('Sorry, you entered wrong choice !'),!.

PRACTICAL 7

/\* multi(N1, N2, R):-R is N1 \* N2.\*/

multi :-

write('Enter 1st number : '),

read(X),

write('Enter 2nd number : '),

read(Y),

Z is X\*Y,

write('Product of '),write(X),write(' and '),write(Y),write(' is '),write(Z),nl,

write('Do you want to continue ? (yes/no) : '),

read(Ch),

cont(Ch).

cont(yes) :- multi.

cont(no) :- !.

cont(Ch) :- write('Sorry, you entered wrong choice !'),!.

PRACTICAL 8

% X IS THE HEAD OF L

%X IS THE MEMBER OF TAIL OF L

member(X,[X|Tail]).

member(X,[Head|Tail]):-member(X,Tail)

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

PRACTICAL 10

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

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).

PRACTICAL 11

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

palind([]):- write('palindrome').

palind([\_]):- write('palindrome').

palind(L) :-

append([H|T], [H], L),

palind(T)

;

write('Not a palindrome').

PRACTICAL 12

sumlist([],0).

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

PRACTICAL 13

evenLength(L):-count(L,C),C1 is C mod 2,C1=0.

oddLength(L):-count(L,C),C1 is C mod 2,C1=1.

count([],0).

count([H|T],C):-count(T,C1),C is C1+1.

PRACTICAL 14

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

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

PRACTICAL 15

maxlist([],0).

maxlist([Head|Rest],Max):-maxlist(Rest,Max2),Head>Max2,Max is Head;

maxlist(Rest,Max2),Head<Max2,Max is Max2.

PRACTICAL 16

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

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

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

PRACTICAL 17

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

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

delete\_nth(N1,T,R).

PRACTICAL 18

merge([],[],[]).

merge([],L,L).

merge(L,[],L).

merge([H1|T1],[H2|T2],[H1|L]):- H1<H2,

merge(T1,[H2|T2],L).

merge([H1|T1],[H2|T2],[H2|L]):- H2<H1,

merge([H1|T1],T2,L).