AI Practical
Q1 to Q18
Using Cut & Fail

Name - Vaibhav Dubey
Roll No. - 5821

1. Write a prolog program to calculate the sum of two numbers.
=>
```
sum(X, Y, Z) :-
   number(X),
   number(Y),
   !,
   Z is X + Y.
sum(_, _, _) :-
   fail.
```

2. Write a Prolog program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.
=>
```
max(X, Y, M) :-
   number(X),
   number(Y),
   !,
   (X >= Y -> M = X ; M = Y).
max(_, _, _) :-
   fail.
```

3. Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N.
=>
```
factorial(0, 1) :-
   !.
factorial(N, F) :-
   N > 0,
   !,
   N1 is N - 1,
   factorial(N1, F1),
   F is N * F1.
factorial(_, _) :-
   fail.
```

4. Write a program in PROLOG to implement generate_fib(N,T) where T represents the Nth term of the fibonacci series.
=>
```
fib(1, 1) :- !.
fib(2, 1) :- !.
fib(N, T) :-
```

```
    N > 2,
    N1 is N - 1,
    N2 is N - 2,
    fib(N1, T1),
    fib(N2, T2),
    T is T1 + T2.
```

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

=>

```
    gcd(X, Y, G) :-
        X =< 0, Y =< 0,
        !,
        fail.
    gcd(X, 0, X) :-
        !.
    gcd(0, Y, Y) :-
        !.
    gcd(X, Y, G) :-
        X >= Y,
        !,
        X1 is X - Y,
        gcd(X1, Y, G).
    gcd(X, Y, G) :-
        X < Y,
        !,
        Y1 is Y - X,
        gcd(X, Y1, G).
```

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

=>

```
power(_, 0, 1) :-
    !.
power(Num, 1, Num) :-
    !.
power(Num, Pow, Ans) :-
    Pow > 0,
    !,
    Pow1 is Pow - 1,
    power(Num, Pow1, Ans1),
    Ans is Ans1 * Num.
power(_, _, _) :-
    fail.
```

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

=>

```
    multi(_, 0, 0) :-
        !.
```

```
multi(0, _, 0) :-
    !.
multi(N1, N2, R) :-
    N1 > 0,
    N2 > 0,
    !,
    N21 is N2 - 1,
    multi(N1, N21, R1),
    R is R1 + N1.
multi(N1, N2, R) :-
    N1 < 0,
    N2 > 0,
    !,
    N11 is -N1,
    multi(N11, N2, R1),
    R is -R1.
multi(N1, N2, R) :-
    N1 > 0,
    N2 < 0,
    !,
    N21 is -N2,
    multi(N1, N21, R1),
    R is -R1.
multi(_, _, _) :-
    fail.
```

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

=>

```
memb(X, [X|_]) :-
    !.
memb(X, [_|T]) :-
    memb(X, T),
    !.
memb(_, _) :-
    fail.
```

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([], L2, L2) :-
    !.
conc([H|T], L2, [H|L3]) :-
    conc(T, L2, L3),
    !.
conc(_, _, _) :-
    fail.
```

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

=>

```
conc([], L2, L2) :-
    !.
conc([H|T], L2, [H|L3]) :-
    conc(T, L2, L3),
    !.
conc(_, _, _) :-
    fail.
```

Q1 to Q10 Output :-

```
?- sum(78,89,X).
X = 167.

?- max(56,34,X).
X = 56 .

?- fact(6,X).
X = 720 .

?- fib(10,X).
X = 55 .

?- gcd(34,56,X).
X = 2 .

?- pow(4,3,X).
X = 64 .

?- multi(6,9,X).
```

```
SWI-Prolog (AMD64, Multi-threaded, version 9.0.3)
File  Edit  Settings  Run  Debug  Help
?- gcd(34,56,X).
X = 2 .

?- pow(4,3,X).
X = 64 .

?- multi(6,9,X).
X = 54 .

?- member(a,[a,b,c,d]).
true .

?- member(f,[a,b,c,d]).
false.

?- con([1,2,3,4],[a,b,c,d],L).
L = [1, 2, 3, 4, a, b, c, d].

?- revl([1,2,3,4,a,b,c],X,[]).
X = [c, b, a, 4, 3, 2, 1].

?-
```

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

=>

```
palindrome(L) :-
    reverse(L, L),
    !.
palindrome(_, _) :-
    fail.
```

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

=>

```
sumlist([], 0) :-
    !.
sumlist([H|T], S) :-
    sumlist(T, ST),
    S is ST + H,
    !.
sumlist(_, _) :-
    fail.
```

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.

=>

```
evenlength([]) :-
    !.
evenlength([_|T]) :-
    oddlength(T),
    !.
evenlength(_, _) :-
    fail.

oddlength([_]) :-
    !.
oddlength([_|T]) :-
    evenlength(T),
    !.
oddlength(_, _) :-
    fail.
```

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.

=>

```
nth_element(1, [H|_], H) :-
    !.
nth_element(N, [_|T], X) :-
    M is N - 1,
    nth_element(M, T, X),
    !.
nth_element(_, [], _) :-
    fail.
nth_element(N, L, _) :-
    length(L, Len),
    N > Len,
    !,
    fail.
```

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

=>

```
maxlist([X], X) :- !.
maxlist([H|T], M) :-
   maxlist(T, N),
   (H > N -> M = H ; M = N),
   !.
maxlist([], _) :-
   fail.
```

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.

=>

```
insert_nth(I, N, L, R) :-
   N > 0,
   N1 is N - 1,
   length(L, Len),
   N =< Len,
   insert_nth_helper(I, N1, L, [], R),
   !.

insert_nth(_, _, L, L) :-
   fail.

insert_nth_helper(I, 0, L, Acc, R) :-
   append(Acc, [I|L], R).

insert_nth_helper(I, N, [H|T], Acc, R) :-
   N > 0,
   N1 is N - 1,
   append(Acc, [H], Acc1),
   insert_nth_helper(I, N1, T, Acc1, R).
```

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.

=>

```
delete_nth(1, [_|T], T) :- !.
delete_nth(N, [H|T], [H|R]) :- N > 1,
                               N1 is N - 1,
                               delete_nth(N1, T, R).
```

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.

=>

```
merge([], L2, L2) :- !.
merge(L1, [], L1) :- !.
merge([H1|T1], [H2|T2], [H1|T]) :- H1 =< H2, !, merge(T1, [H2|T2], T).
```

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

Q11 to Q18 Output :-