Basic arithmetic and logic problems

1.Maximum of Two Numbers :

```
max(X, Y, X) :- X >= Y.
max(X, Y, Y) :- X < Y.
o/p- ?- max(5, 8, Max). → Max = 8
```

2. Factorial of a Number (m!) :

```
factorial(0, 1).
factorial(N, F) :-
  N > 0,
  N1 is N - 1,
  factorial(N1, F1),
  F is N * F1.
o/p- ?- factorial(5, F). → F = 120
```

3. Sum of Two Numbers (m + n) :

```
sum(M, N, R) :-
  R is M + N.
o/p- ?- sum(7, 3, R). → R = 10
```

4. Sum of First N Natural Numbers :

```
sum_natural(0, 0).
sum_natural(N, Sum) :-
  N > 0,
  N1 is N - 1,
  sum_natural(N1, S1),
  Sum is S1 + N.
o/p- ?- sum_natural(5, S). → S = 15
```

5. GCD of Two Numbers :

```
gcd(X, 0, X).
gcd(X, Y, G) :-
  Y > 0,
  R is X mod Y,
  gcd(Y, R, G).
o/p- ?- gcd(48, 18, G). → G = 6
```

6. Sum of Digits of a 3-digit Number :

```
sum_digits(N, Sum) :-
  N >= 100,
  N =< 999,
  A is N // 100,
  B is (N // 10) mod 10,
  C is N mod 10,
  Sum is A + B + C.
o/p- ?- sum_digits(456, S). → S = 15
```

7. Nth Fibonacci Number :

```
 fibonacci(0, 0).
fibonacci(1, 1).
fibonacci(N, F) :-
  N > 1,
  N1 is N - 1,
  N2 is N - 2,
  fibonacci(N1, F1),
  fibonacci(N2, F2),
  F is F1 + F2.
o/p- ?- fibonacci(6, F). → F = 8
```

8. Sum of Digits of an N-Digit Number :

```
sum_digits(0, 0).
sum_digits(N, Sum) :-
  N > 0,
  D is N mod 10,
  N1 is N // 10,
  sum_digits(N1, RestSum),
  Sum is D + RestSum.
o/p - ?- sum_digits(12345, S). -> S = 15.
```

9. Check if a Number is Even or Odd :

```
even(N) :- N mod 2 =:= 0.
odd(N) :- N mod 2 =:= 1.
o/p - even(4). -> true.
      odd(7). -> true.
```

10. Check if a Number is Prime :

```
is_prime(2).
is_prime(N) :-
  N > 2,
  \+ has_factor(N, 2).

has_factor(N, F) :-
  N mod F =:= 0.
has_factor(N, F) :-
  F * F < N,
  F1 is F + 1,
  has_factor(N, F1).
o/p- is_prime(2). -> true.
```

11. Find the Power (M^N)

```
power(_, 0, 1).
power(M, N, R) :-
  N > 0,
  N1 is N - 1,
  power(M, N1, R1),
  R is M * R1.
o/p-  power(2, 3, R). -> R = 8.
```

12. Find LCM of Two Numbers :

```
gcd(X, 0, X).
gcd(X, Y, G) :- Y > 0, R is X mod Y, gcd(Y, R, G).

lcm(X, Y, L) :-
  gcd(X, Y, G),
  L is (X * Y) // G.
o/p -  lcm(4, 6, L). -> L = 12.
```

13. Check if a Number is a Palindrome :

```
is_number_palindrome(N) :-
  number_chars(N, L),
  reverse(L, L).
o/p- is_number_palindrome(121). -> true.
```

List Operations

14. Remove Nth Item from a List :

```
remove_nth(1, [_|T], T).
remove_nth(N, [H|T], [H|R]) :-
  N > 1,
  N1 is N - 1,
  remove_nth(N1, T, R).
o/p - ?- remove_nth(3, [a, b, c, d], R).
R = [a, b, d].
```

15. Palindrome Check (List) :

```
palindrome(L) :- reverse(L, L).
reverse([], []).
reverse([H|T], R) :-
  reverse(T, RT),
  append(RT, [H], R).
o/p- ?- palindrome([r, a, c, e, c, a, r]).
true.
```

16. Palindrome Check (List) :

```
maxlist([X], X).
maxlist([H|T], Max) :-
  maxlist(T, TempMax),
  Max is max(H, TempMax).
o/p- ?- maxlist([3, 7, 2, 9, 1], M). -> M = 9.
```

17. Sum of Elements in a List (sumlist/2) :

```
sumlist([], 0).
sumlist([H|T], Sum) :-
  sumlist(T, Rest),
  Sum is H + Rest.
o/p -  ?- sumlist([1, 2, 3, 4], S).-> S = 10.
```

18. Reverse a List (reverse/2) :

```
reverse([], []).
reverse([H|T], R) :-
  reverse(T, RT),
  append(RT, [H], R).
o/p - ?- reverse([a, b, c, d], R).
R = [d, c, b, a].
```

19. Insert Element in a List (Beginning, End, Any Position) :

```
insert_first(E, L, [E|L]).

insert_last(E, [], [E]).
insert_last(E, [H|T], [H|R]) :-
  insert_last(E, T, R).

insert_at(E, 1, L, [E|L]).
insert_at(E, N, [H|T], [H|R]) :-
  N > 1,
  N1 is N - 1,
  insert_at(E, N1, T, R).
o/p –
 ?- insert_first(x, [a, b, c], R).
R = [x, a, b, c].
?- insert_last(x, [a, b, c], R).
R = [a, b, c, x].
?- insert_at(x, 2, [a, b, c], R).
R = [a, x, b, c].
```

20. Check if a List has Even or Odd Length :

```
even_length([]).
even_length([_,_|T]) :- even_length(T).

odd_length([_]).
odd_length([_,_|T]) :- odd_length(T).
o/p - even_length([a, b, c, d]).-> true.
```

21. Append Two Lists :

```
append_list([], L, L).
append_list([H|T], L, [H|R]) :-
  append_list(T, L, R).
o/p -  ?- append_list([a, b], [c, d], R).
R = [a, b, c, d].
```

22. m! / (m + n) :

```
factorial(0, 1).
factorial(M, F) :-
  M > 0,
  M1 is M - 1,
  factorial(M1, F1),
  F is M * F1.

% m! / (m + n)
calculate(M, N, Result) :-
  factorial(M, F),
  Sum is M + N,
  Sum =\= 0,       % Avoid division by zero
  Result is F / Sum.
o/p - calculate(5, 3, R). -> R = 15.0.
```

23. $a^2 - b^2 + c^2$:

```
calculate_expr(A, B, C, Result) :-
  A2 is A * A,
  B2 is B * B,
  C2 is C * C,
  Result is A2 - B2 + C2.
o/p -  ?- calculate_expr(3, 2, 4, R).-> R = 21.
```

24. Delete First and Last Element(list) :

```
delete_first([_|T], T).
delete_last([_], []).
delete_last([H|T], [H|R]) :-
  delete_last(T, R).
o/p -?- delete_first([a, b, c, d], R).
R = [b, c, d].
?- delete_last([a, b, c, d], R).
R = [a, b, c].
```

25. Check if Two Lists Are Equal :

```
list_equal([], []).
list_equal([H1|T1], [H2|T2]) :-
  H1 =:= H2,
  list_equal(T1, T2).
o/p -?- list_equal([1,2,3], [1,2,3]).-> true.
```

26. Find Intersection of Two Lists :

```
intersection([], _, []).
intersection([H|T], L2, [H|R]) :-
  member(H, L2),
  intersection(T, L2, R).
intersection([H|T], L2, R) :-
  \+ member(H, L2),
  intersection(T, L2, R).
o/p -  ?- intersection([1,2,3,4], [3,4,5], R).
R = [3, 4].
```

27. Find Union of Two Lists (No Duplicates)

```
union([], L, L).
union([H|T], L, R) :-
  member(H, L),
  union(T, L, R).
union([H|T], L, [H|R]) :-
  \+ member(H, L),
  union(T, L, R).
o/p -  ?- union([1,2,3], [3,4,5], R).
R = [1, 2, 3, 4, 5].
```

28. Remove All Occurrences of an Element from a List :

```
remove_all(_, [], []).
remove_all(X, [X|T], R) :-
  remove_all(X, T, R).
remove_all(X, [H|T], [H|R]) :-
  X \= H,
  remove_all(X, T, R).
o/p -  ?- remove_all(2, [1,2,2,3,2], R).
R = [1, 3].
```

29. Find the Second Largest Element in a List :

```
maxlist([X], X).
maxlist([H|T], Max) :-
  maxlist(T, Temp),
  Max is max(H, Temp).

remove_first(_, [], []).
remove_first(X, [X|T], T) :- !.
remove_first(X, [H|T], [H|R]) :-
  remove_first(X, T, R).

second_largest(L, Second) :-
  maxlist(L, Max),
  remove_first(Max, L, L1),
  maxlist(L1, Second).
O/p -  ?- second_largest([10, 20, 5, 8], S).
S = 10.
```

30. Armstrong Number Checker:

```
% split number into digits
digits(0, []) :- !.
digits(N, [D|Rest]) :-
  D is N mod 10,
  N1 is N // 10,
  digits(N1, Rest).

% compute sum of digits^N
sum_powers([], _, 0).
sum_powers([D|T], Power, Sum) :-
  sum_powers(T, Power, Rest),
  Sum is D**Power + Rest.

% main predicate
is_armstrong(N) :-
  digits(N, Digits),
  length(Digits, Len),
  sum_powers(Digits, Len, Sum),
  Sum =:= N.
O/p - is_armstrong(153). -> true.
```

31.Family Tree in Prolog

Facts

```
% Gender
male(john).
male(mike).
male(tom).

female(mary).
female(susan).
female(anna).

% Parent relationships
parent(john, mike).
parent(mary, mike).
parent(john, anna).
parent(mary, anna).
parent(mike, tom).
parent(susan, tom).
```

Basic Relations:

```
child(X, Y) :- parent(Y, X).
grandparent(X, Z) :- parent(X, Y), parent(Y,
Z).
mother(X, Y) :- parent(X, Y), female(X).
predecessor(X, Y) :- parent(X, Y).
predecessor(X, Y) :- parent(X, Z),
predecessor(Z, Y).
aunt(A, X) :-
  parent(P, X), parent(G, P),  parent(G, A),
  female(A), A \= P.
o/p :
?- child(mike, john). -> true.
?- grandparent(john, tom). -> true
?- mother(mary, anna). -> true
?- aunt(anna, tom). -> true
```

Index