1. **Write a prolog program to find factorial of a given number.**

```
predicates
    start
    find_factorial(real,real)

goal
    clearwindow,
    start.

clauses
    start:-
        write("Enter non negative number = "),
        readreal(Num),
        Result = 1.0,
        find_factorial(Num,Result).

    find_factorial(Num,Result):-
        Num <> 0,
        NewResult = Num * Result,
        NewNum = Num - 1,
        find_factorial(NewNum,NewResult).

    find_factorial(_,Result):-
        write("Factorial = ",Result),nl.
```

2. **Write a prolog program to find maximum number from a list.**

```
domains
    list = integer*
    Max = integer
predicates
    maximum_no(list,integer)
clauses
    maximum_no([],Max):-
        write("Maximum No in List is:: ",Max),nl.
    maximum_no([H|T],Max):-
        H>Max,
        N = H,
        maximum_no(T,N).
    maximum_no(L,Max):-
        maximum_no(L,Max).
```

3. Write a prolog program to find sum of all the numbers of a list.

```
domains
   list=integer*
predicates
   findsum(list)
   sum(list,integer)
clauses
   findsum(L):-
      sum(L,Sum),
      write("\nSum Of Given List : ",Sum).
   sum([],0).
      sum([X|Tail],Sum):-
      sum(Tail,Temp),
      Sum=Temp+X.
```

```
OUT PUT
=======

Goal: findsum([1,2,3,4,5])

Sum Of Given List : 15

Yes

-------------------------------------

Goal: findsum([])

Sum Of Given List : 0

Yes

-------------------------------------

Goal: findsum([1,2,3,4,5,6,7,8,9,10])

Sum Of Given List : 55

Yes
```

4. **Write a prolog program to reverse the given list.**

```
domains
    l = integer*

predicates
    reverse_list(l,l)
    reverse(l,l,l)

clauses
    reverse_list(IN,OUT) :-
    reverse(IN,[],OUT).
    reverse([],IN,IN).
    reverse([Head|Tail],List1,List2) :-
    reverse(Tail,[Head|List1],List2).
```

5. **Write a Prolog program to merge two sequentially ordered (ascending) lists into one ordered list.**

```
domains
    x = integer
    l = integer*
predicates
    mergelist(l,l,l)
clauses
    mergelist([],[],[]).
    mergelist([X],[],[X]).
    mergelist([],[Y],[Y]).
    mergelist([X|List1],[Y|List2],[X|List]) :-
        X <= Y,!,
    mergelist(List1,[Y|List2],List).
    mergelist([X|List1],[Y|List2],[Y|List]) :-
    mergelist([X|List1],List2,List).

Output :

Goal: mergelist([1,3,5],[2,4,6],List)
List=[1,2,3,4,5,6]
1 Solution

Goal: mergelist([-1,1,4,5],[-3,0,2,3,5],List)
List=[-3,-1,0,1,2,3,4,5,5]
1 Solution
```

6. **Write a Prolog program for finding a set, which is result of the intersection of the two given sets.**

```
clauses:
     intersectionTR(_, [], []).
     intersectionTR([], _, []).
     intersectionTR([H1|T1], L2, [H1|L]):-
     member(H1, L2),
     intersectionTR(T1, L2, L), !.
     intersectionTR([_|T1], L2, L):-
     intersectionTR(T1, L2, L).

     intersection(L1, L2):-
                    intersectionTR(L1, L2, L),
                    write(L).

unionTR([], [], []).
unionTR([], [H2|T2], [H2|L]):-
     intersectionTR(T2, L, Res),
     Res = [],
     unionTR([], T2, L),
     !.
unionTR([], [_|T2], L):-
     unionTR([], T2, L),
     !.

unionTR([H1|T1], L2, L):-
     intersectionTR([H1], L, Res),
     Res \= [],
     unionTR(T1, L2, L).

unionTR([H1|T1], L2, [H1|L]):-
     unionTR(T1, L2, L).

union(L1, L2):-
     unionTR(L1, L2, L),
     write(L).

Goal:
     intersect([1,3,5,2,4] ,[6,1,2]).

Output:
     [1,2]
     Yes
```

7. Write a prolog program to check whether a number is a member of given list or not.

```
domains

   list=integer*


predicates
   findnum(integer,list)
clauses
   findnum(X,[]):-
      write("\nNumber Is Not Found").
   findnum(X,[X|Tail]):-
      write("\nNumber Is Found").
   findnum(X,[Y|Tail]):-
      findnum(X,Tail).
```

```
OUT PUT
=======

Goal: findnum(3,[1,2,3,4,5])

Number Is Found

Yes

---------------------------

Goal: findnum(6,[1,2,3,4,5])

Number Is Not Found

Yes

---------------------------

Goal: findnum(2,[1,2,2,1])

Number Is Found

Yes
```

8. Write a prolog program to concatenating of two lists.

```
domains
    list=symbol*

predicates
    con(list,list,list)

clauses
    con([],L1,L1).
    con([X|Tail],L2,[X|Tail1]):-
     con(Tail,L2,Tail1).

OUT PUT
=======
Goal: con([a,b,c],[d,e],ConcatList)
ConcatList=["a","b","c","d","e"]
1 Solution
```

9. Write a prolog program to delete an element from a list.

```
domains
    list=symbol*

predicates
    del(symbol,list,list)

clauses
    del(X,[X|Tail],Tail).
    del(X,[Y|Tail],[Y|Tail1]):-
    del(X,Tail,Tail1).

OUT PUT
=======
Goal: del(c,[a,b,c,d,e],NewList)
NewList=["a","b","d","e"]
1 Solution
--------------------------------
Goal: del(a,[b,a,c,a],L)
L=["b","c","a"]
L=["b","a","c"]
2 Solutions
```