Write a program to find the sum of elements of a list using Prolog.

```
domains
   x = integer
   l = integer*

predicates
   sum(l,x)

clauses
   sum([],0).

   sum([X|List],S) :-
      sum(List,S1),
      S = X + S1.
```

Write a program to solve Tower of Hanoi problem using Prolog.

```
DOMAINS
  POLE = SYMBOL
PREDICATES
  hanoi
  hanoi(INTEGER)
  move(INTEGER,POLE,POLE,POLE)
  inform(POLE,POLE)

CLAUSES
  hanoi:-hanoi(5).
  hanoi(N) :- move(N, left, middle, right).
  move(1, A, _, C) :- inform(A, C), !.
  move(N, A, B, C) :-
        N1 is N-1,
        move(N1, A, C, B),
        inform(A, C), !,
        move(N1, B, A, C).
  inform(Loc1, Loc2) :-  write("Move a disk from ", Loc1, " to ", Loc2),nl, !.
```

Write a program to solve N-Queens problem using Prolog.

```
domains
        cell=c(integer,integer)
        list=cell*
        int_list=integer*
predicates
        solution(list)
        member(integer,int_list)
        nonattack(cell,list)
clauses
        solution([]).
```

```
solution([c(X,Y)|Others]):-
solution(Others),
member(Y,[1,2,3,4,5,6,7,8]),
nonattack(c(X,Y),Others).
nonattack(_,[]).
nonattack(c(X,Y),[c(X1,Y1)|Others]):-
Y<>Y1,
Y1-Y<>X1-X,
Y1-Y<>X-X1,
nonattack(c(X,Y),Others).
member(X,[X|_]).
member(X,[_|Z]):-
member(X,Z).
```

Write a program to solve Travelling Salesman Problem using Prolog

```
domains
town = symbol
distance = integer
predicates
nondeterm road(town,town,distance)
nondeterm route(town,town,distance)
clauses
road("surat","bardoli",200).
road("mandvi","surat",300).
road("bardoli","mandvi",100).
road("vyara","bardoli",120).
road("surat","vyara",130).
route(Town1,Town2,Distance):-
road(Town1,Town2,Distance).
route(Town1,Town2,Distance):-
road(Town1,X,Dist1),
route(X,Town2,Dist2),
Distance=Dist1+Dist2,!.
```

(i) Write a PROLOG Program to Find the Sum of first N natural numbers.

```
predicates
   add_upto(integer, integer)

clauses
  add_upto(1, 1)
  add_upto(N, Result) if
          N>=2,
          N_1=N-1,
          add_upto(N_1, Res),
          Result=Res+N.
```

```
goal
   add_upto(3, Result)
      Result=6
```

(ii) Prolog program to concatenate two lists giving third list

```
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
```

(i) Find factorial

```
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.
```

(ii) Prolog program to find the nth element of a list

```
domains
   x = integer
   l = integer*

predicates
   find(l,x)

clauses
   find([],N) :-
      write(\"There is no such element in the list\"),nl.
   find([Element|List],1) :-
      write(\"The element is \",Element),nl.
   find([Element|List],N) :-
      N1 = N-1,
      find(List,N1).
```

Output :

```
Goal: find([1,2,3,4],3)
The element is 3
Yes
Goal: find([1,2,3,4],0)
There is no such element in the list
Yes
```

3. PROLOG Program to Find GCD of Two Numbers.

```
predicates
   gcd(integer, integer, integer)

clauses
   gcd(M, O, M).
   gcd(M, N, Result):-
      Rem=M mod N,
      gcd(N, Rem, Result).
```

Output:

```
goals:
   gcd(6, 4, Result)
      Result=2
```

(i)Find last element in a given list

```
domains
   list=symbol*

predicates
   last(list)

clauses
   last([X]):-
      write(\"\\nLast element is : \"),
      write(X).
   last([Y|Tail]):-
      last(Tail).
```

OUT PUT

```
=======
Goal: last([a,b,c,d,e])
Last element is : e
Yes
```

10.find the number positive or not

```
predicates
        number(integer)

clauses
        number(X):- X>0
                write("positive").
        number(-):- write("nagative")
```