p1
safe_queens([B,D],A,2)
A #\= B
abs(A-B)#\= 1
D1 #= 1+1 -> 2

safe_queens([C,D],A,2)
A #\= C
abs(A-C)#\= 2
D1 #= 2+1 -> 3

safe_queens([D],A,3)
A #\= D
abs(A-D)#\= 3
D1 #= 3+1 -> 4
safe_queens([],A,4)

safe_queens([C,D],B,1)
B #\= C
abs(B-C)#\= 1
D1 #= 1+1 -> 2

safe_queens([D],B,2)
B #\= D
abs(B-D)#\= 2
D1 #= 2+1 -> 3
safe_queen([],B, 3)

safe_queens([D],C,1)
C #\= D
abs(C-D)#\= 1
D1 #= 1+1 -> 2
safe_queens([],C,2)

Constraint :

A #\= B
abs(A-B)#\= 1
A #\= C
abs(A-C)#\= 2
A #\= D
abs(A-D)#\= 3
B #\= C
abs(B-C)#\= 1
B #\= D
abs(B-D)#\= 2
C #\= D

```
abs(C-D)#\= 1


%9 limit the number of rows to 9 and each row with length 9.
%10 limit value from 1-9 only
%11 limit the values , so all value in a row are unique
%12 create a list that contains the list columns.
%13 limit the value so all values in a column are unique
%14 each row are named with a letter
%15 solver by doing 3 x 3 sudoku grid instead of 9 x 9

%18 split into first 3 value and the rest
%19 merge the 3 x 3 grid so 9 x 9 grid in the end
%20 all values in row/columns is unique in a grid
%21 recursively call blocks to ch.

% Example : You meet 3 inhabitants. A says: "All of us are knaves."
%           B says: "Exactly one of us is a knight."

example_knights(4, Ks) :-
        Ks = [A,B,C],
        sat(A=:=(~A * ~B * ~C)),
        sat(B=:=card([1],Ks)).

% Example : A says: "B is a knave."
%           B says: "A and C are of the same kind."
%           What is C?

example_knights(5, [A,B,C]) :-
        sat(A=:= ~B),
        sat(B=:=(A=:=C)).

% Example : A says: "I am a knave, but B isn't."

example_knights(2, [A,B]) :-
        sat(A=:=(~A * B)).




p2
:- use_module(library(clpfd)).

rev([H|L], Ret) :- revHelp([H|L],[],Ret).
```

```
        revHelp([],Abb,Abb).
        revHelp([H|L],Abb,Ret) :-revHelp([L],[H|Abb],Ret).

crypt1([H1|L1],[H2|L2],[H3|L3],L4) :-
   L4 ins 0..9, all_different(L4),
   H1 #\= 0,
   H2 #\= 0,
   H3 #\= 0,
   rev([H1|L1], RevL1),
   rev([H2|L2], RevL2),
   rev([H3|L3], RevL3),
   crypt1_Help(RevL1, RevL2, RevL3, 0).

        crypt_Help([],[],[CZ],CZ).
   crypt_Help([], [H|L], [H2|L2], CZ) :-
   H2 #= (H + CZ) mod 10,
   H2 + (CO*10) #= (H + CZ),
   crypt_Help([],L,L2,CO).

        crypt_Help([H|L],[],[H2|L2],CZ) :-
   H2 #= (H + CZ) mod 10,
   H2 + (CO*10) #= (H + CZ),
   crypt_Help([L],[],L2,CO).

        crypt_Help([H1|L1],[H2|L2],[H3|L3], CZ) :-
   H3 #= (H1 + H2 + CZ) mod 10,
   H3 + (CO*10) #= (H1 + H2 + CZ),
   crypt_Help(L1,L2,L3,CO).
```