```
% load files

:- consult('../gv.pl').

% random crypto problem generation

establishCryptoProblemParameters :-
        declare(lo, 0),
        declare(hi, 15).

generateRandomCryptoNumber(R) :-
        valueOf(lo,Lo),
        valueOf(hi,Hi),
        HiPlus1 is Hi + 1,
        random(Lo,HiPlus1,R).

generateRandomCryptoProblem :-
        generateRandomCryptoNumber(N1),
        generateRandomCryptoNumber(N2),
        generateRandomCryptoNumber(N3),
        generateRandomCryptoNumber(N4),
        generateRandomCryptoNumber(N5),
        generateRandomCryptoNumber(G),
        addCryptoProblemToKnowledgeBase(N1,N2,N3,N4,N5,G).

addCryptoProblemToKnowledgeBase(N1,N2,N3,N4,N5,G) :-
        retract(problem(_,_)),
        assert(problem(numbers(N1,N2,N3,N4,N5),goal(G))).
addCryptoProblemToKnowledgeBase(N1,N2,N3,N4,N5,G) :-
        assert(problem(numbers(N1,N2,N3,N4,N5),goal(G))).

%display the problem -- assuming that it has been internalized
%

displayProblem :-
        problem(numbers(N1,N2,N3,N4,N5),goal(G)),
        write('Problem: numbers = (') ,
        write(N1), write(','),
        write(N2), write(','),
        write(N3), write(','),
        write(N4), write(','),
        write(N5), write('} and goal = '),
        write(G), nl.

% random crypto problem generation demo
%

demo :-
        generateRandomCryptoProblem,
        displayProblem.

% random crypto problem generation
%

genome :-
        generateRandomCryptoProblem,
        displayProblem.

generate(1) :- genome.
generate(N) :-
        genome,
        M is N-1,
        generate(M).
```

```
%initialization
%

:- establishCryptoProblemParameters.
```