```
%crypto.pro: a program to generate and display crypto problems

%instantiating gv.pro
:- consult('gv.pro').

% this program will estimate a range constant on the crypto problems to
% be answered.

% Write a method to place a random crypto problem in the KB by establishing
% a global variable.


%establishing the range for the crypto problem generations
establish_crypto_problem_parameters :-
declare(lo,0),
declare(hi,9).
:- establish_crypto_problem_parameters.


%generates random numbers in between range of numbers

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

%generates the numbers for the crypto problems
generate_random_crypto_problem :-
   generate_random_crypto_number(N1),
   generate_random_crypto_number(N2),
   generate_random_crypto_number(N3),
   generate_random_crypto_number(N4),
   generate_random_crypto_number(N5),
   generate_random_crypto_number(G),
   add_crypto_problem_to_KB(N1,N2,N3,N4,N5,G).

%adds crypto problems to the knowledge base
add_crypto_problem_to_KB(N1,N2,N3,N4,N5,G):-
    undeclare(crypto_problem),
    declare(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),goal(G))).
add_crypto_problem_to_KB(N1,N2,N3,N4,N5,G):-
```

```
    declare(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),goal(G))).


%generates the structure of the crypto problem

generate_one :-
    generate_random_crypto_problem,
    write(crypto_problem),write(‘ --> ‘),binding(crypto_problem, Crypto_problem),
    write(Crypto_problem), nl.


generate(1) :-
    generate_one.

generate(N) :-
    generate(1),
    NM1 is N-1,
    generate(NM1).
```