

## v4/crypto.pro KB

```
% FILE: crypto.pro
% TYPE: Prolog Source
% Line: Crypto problem generation and solution exhaustive search
% DATE: November 6, 2015
```

```
:-consult('gv.pro').
:-consult('combosets.pro').
```

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

```
:-establishCryptoProblemParameters.
```

```
generateRandomCryptoNumber(R) :-
valueOf(lo,Lo),
valueOf(hi,Hi),
Hip is Hi + 1, random(Lo,Hip,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) :-
eraseProblem,
assert(problem(numbers(N1,N2,N3,N4,N5),goal(G))).
```

```
eraseProblem :-
retract(problem(_, _)),
fail. eraseProblem.
```

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.

crypto(N1,N2,Goal,ex(N1,+,N2)) :-  
Goal is (N1 + N2).

crypto(N1,N2,Goal,ex(N1,\*,N2)) :-  
Goal is (N1 \* N2).

crypto(N1,N2,Goal,ex(N1,-,N2)) :-  
Goal is (N1 - N2).

crypto(N1,N2,Goal,ex(N2,-,N1)) :-  
Goal is (N2 - N1).

crypto(N1,N2,Goal,ex(N1/,N2)) :-  
N2 > 0, Goal is (N1 / N2).

crypto(N1,N2,Goal,ex(N2/,N1)) :-  
N1 > 0, Goal is (N2 / N1).

crypto(N1,N2,N3,G,Expr) :-  
combos(set(N1,N2,N3),  
combo(A,B),extras(C)),  
crypto(A,B,SG,SGE),  
crypto(C,SG,G,UGE),  
substitute(SGE,SG,UGE,Expr).

crypto(N1,N2,N3,N4,G,Expr) :-  
combos(set(N1,N2,N3,N4),  
combo(A,B),extras(C,D)),  
crypto(A,B,SG,SGE),  
crypto(C,D,SG,G,UGE),  
substitute(SGE,SG,UGE,Expr).

```

crypto(N1,N2,N3,N4,N5,G,Expr) :-
combos(set(N1,N2,N3,N4,N5),
combo(A,B),extras(C,D,E)),
crypto(A,B,SG,SGE),
crypto(C,D,E,SG,G,UGE),
substitute(SGE,SG,UGE,Expr).
substitute(New,Old,ex(Old,O,Z),ex(New,O,Z)).
substitute(New,Old,ex(X,O,Old),ex(X,O,New)).

```

```

substitute(New,Old,ex(X,O,Z),ex(Q,O,Z)) :-
substitute(New,Old,X,Q).

```

```

substitute(New,Old,ex(X,O,Z),ex(X,O,Q)) :-
substitute(New,Old,Z,Q).

```

```

displaySolution :-
write('Solution: '),
solution(S),
displayResult(S), nl.
displaySolution.

```

```

displayResult(ex(A,O,B)) :-
number(A),
number(B),
write('('), write(A), write(' '), write(O), write(' '), write(B), write(')').

```

```

displayResult(ex(A,O,B)) :-
number(A),
B = ex(A1,O1,B1),
write('('), write(A), write(' '), write(O), write(' '), displayResult(ex(A1,O1,B1)), write(')').

```

```

displayResult(ex(A,O,B)) :-
number(B),
A = ex(A1,O1,B1),
write('('), displayResult(ex(A1,O1,B1)), write(' '), write(O), write(' '), write(B), write(')').

```

```

displayResult(ex(A,O,B)) :-
A = ex(A1,O1,B1),

```

```
B = ex(A2,O2,B2),  
write(' '), displayResult(ex(A1,O1,B1)), write(' '), write(O), write(' '),  
displayResult(ex(A2,O2,B2)), write(' ')).
```

```
solveProblemDecompositionally :-  
problem(numbers(N1,N2,N3,N4,N5),goal(G)),  
crypto(N1,N2,N3,N4,N5,G,Expr),  
recordSolution(Expr).
```

```
solveProblemDecompositionally :-  
write('No solution to this one!'), nl.
```

```
recordSolution(Expr) :-  
eraseSolution,  
assert(solution(Expr)).
```

```
eraseSolution :-  
retract(solution(_)),  
fail.  
eraseSolution.
```

```
demo :-  
generateRandomCryptoProblem,  
displayProblem,  
solveProblemDecompositionally,  
displaySolution.
```

```
demo(0).  
demo(N) :-  
demo,  
K is N -1,  
demo(K).
```

```
solve(numbers(N1,N2,N3,N4,N5),goal(G)) :-  
establishCryptoProblem(numbers(N1,N2,N3,N4,N5),goal(G)),  
displayProblem,  
solveProblemDecompositionally,  
displaySolution.
```

```
establishCryptoProblem(numbers(N1,N2,N3,N4,N5),goal(G)) :-  
addCryptoProblemToKnowledgeBase(N1,N2,N3,N4,N5,G).
```