

```
% FILE: crypto.pro of directory xsolve
% TYPE: Prolog source
% LINE: Crypto - exhaustive solver random problems (refactored)
% DATE: Fall 2011
```

```
%-----
% load some files
```

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

```
%-----
% random crypto problem generation
```

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

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

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

```
displayProblem :-
    valueOf(problem,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.
```

```
%-----
% internalize the problem
```

```
internalizeProblem :-
    valueOf(problem,problem(numbers(N1,N2,N3,N4,N5),goal(G))),
    eraseProblemBindings,
    eraseProblem,
    eraseSolution,
    declare(problem,problem(numbers(N1,N2,N3,N4,N5),goal(G))),
    declare(n1,N1),
    declare(n2,N2),
    declare(n3,N3),
    declare(n4,N4),
    declare(n5,N5),
    declare(g,G).
```

```

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

eraseSolution :-
    undeclare(solution),
    fail.
eraseSolution.

eraseProblemBindings :-
    undeclare(n1),
    undeclare(n2),
    undeclare(n3),
    undeclare(n4),
    undeclare(n5),
    undeclare(g),
    fail.
eraseProblemBindings.

%-----
% solve the problem decompositionaly -- assuming internalization

solveProblemDecompositionally :-
    valueOf(problem,problem(numbers(N1,N2,N3,N4,N5),goal(G))),
    crypto(N1,N2,N3,N4,N5,G,Expression),
    declare(solution,solution(Expression)).

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).

%-----
% key substitution code

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).

%-----
% display the solution -- assuming that it has been solved

displaySolution :-

```

```

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

%-----
% display the result -- assuming the problem has been "solved"

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(A), 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(' ')).

%-----
% demo

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

%-----
% establish a particular crypto problem

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

%-----
% crypto problem solver

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

%-----
% initialization

:- establishCryptoProblemParameters.

```