```
%load files giving perms,combos,exhaustive crypto up to 5, arithmetic
%:-consult('~/Documents/CSC366/Assignments/assignment03/gv1.pro').
:-consult('~/Documents/CSC366/Assignments/sets/combosets.pro').
:-consult('~/Documents/CSC366/Assignments/crypto/v1/crypto.pro').
:-consult('~/Documents/CSC366/Assignments/crypto/v3/crypto.pro').

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

%solve internalized problems
solveProblemDecompositionally :-
        getProblemFromKnowledgeBase(N1,N2,N3,N4,N5,G),
        crypto(N1,N2,N3,N4,N5,G,Expression),
        addCryptoSolutionToKB(Expression).
solveProblemDecompositionally :-
        write('No solution to this one!'),nl.

getProblemFromKnowledgeBase(N1,N2,N3,N4,N5,G) :-
        problem(numbers(N1,N2,N3,N4,N5),goal(G)).

addCryptoSolutionToKB(Expression) :-
        retract(solution(_)),
        assert(solution(Expression)).
addCryptoSolutionToKB(Expression) :-
        assert(solution(Expression)).

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(' )').

%solves a random problem
solve(random) :-
        generateRandomCryptoProblem,
        displayProblem,
        solveProblemDecompositionally,
        displaySolution.
solve(numbers(N1,N2,N3,N4,N5),goal(G)) :-
        establishSpecificCryptoProblem(N1,N2,N3,N4,N5,G),
        displayProblem,
```

```
        solveProblemDecompositionally,
        displaySolution.

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