```
:- consult(’../gv.pl’).
:- consult(’../combosets.pl’).

% SEGMENT 1: RANDOM CRYPTO PROBLEM GENERATION AND STORE IT IN
% THE FORM: problem(numbers(N1,N2,N3,N4,N5),goal(G))
establishCryptoProblemParameters :-
        declare(lo,0),
        declare(hi,9).

:- 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.

% SEGMENT 2: THE EXHAUSTIVE CRYPTO PROBLEM SOLVER, ORDERS 2, 3,
% THE PROCESSING IS DONE VIA PARAMETERS RATHER THAN BY MEANS OF
% MANIPULATIONS.
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).

% SEGMENT 3: CODE TO DISPLAY THE RESULT OF SOLVING THE PROBLEM
displaySolution :-
        solution(S),
        displayResult(S),
        eraseSolution,
        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(’ )’).

% SEGMENT 4: CODE TO SOLVE THE CRYPTO PROBLEM USING EXHAUSTIVE
% DECOMPOSITION -- ASSUMING THE PROBLEM HAS BEEN INTERNALIZED
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.

% SEGMENT 5: PROGRAM TO DEMO THE GENERATION AND SOLVING O
% CRYPTO PROBLEM OF ORDER 5 WITH NUMBERS IN THE 0..9 RANGE
demo :-
        generateRandomCryptoProblem,
        displayProblem,
        solveProblemHeuristically,
        displaySolution.
        demo(0).
        demo(N) :-
        demo,
        K is N - 1,
        demo(K).

% SEGMENT 6: PROGRAM TO SOLVE A SPECIFIC CRYPTO PROBLEM OF ORDER
% WITH NUMBERS IN THE 0..15 RANGEre
establishCryptoProblem(numbers(N1,N2,N3,N4,N5),goal(G)) :-
        addCryptoProblemToKnowledgeBase(N1,N2,N3,N4,N5,G).

doubleton :-
        problem(numbers(N1,N2,N3,N4,N5),_),
        combos(set(N1,N2,N3,N4,N5),combo(A,B),_),
        A=B.
doubleton(doubleton(A,B),rest(C,D,E)) :-
        problem(numbers(N1,N2,N3,N4,N5),_),
        combos(set(N1,N2,N3,N4,N5),combo(A,B),extra(C,D,E)),
        A=B.

addition(addition(A,B),rest(C,D,E)) :-
        problem(numbers(N1,N2,N3,N4,N5),goal(G)), combos(set(N1,N2,N3,N4,N5),combo(A,B)
,extra(C,D,E)), G is A + B.

subtraction(subtraction(A,B),rest(C,D,E)) :-
        problem(numbers(N1,N2,N3,N4,N5),goal(G)),
        combos(set(N1,N2,N3,N4,N5),combo(A,B),extra(C,D,E)),
        G is A - B.
subtraction(subtraction(B,A),rest(C,D,E)) :-
        problem(numbers(N1,N2,N3,N4,N5),goal(G)),
        combos(set(N1,N2,N3,N4,N5),combo(A,B),extra(C,D,E)),
        G is B - A.

multiplication(multiplication(A,B),rest(C,D,E)) :-
        problem(numbers(N1,N2,N3,N4,N5),goal(G)),
        combos(set(N1,N2,N3,N4,N5),combo(A,B),extra(C,D,E)),
        G is A * B.

division(division(A,B),rest(C,D,E)) :-
        problem(numbers(N1,N2,N3,N4,N5),goal(G)),
        combos(set(N1,N2,N3,N4,N5),combo(A,B),extra(C,D,E)),
        not(B=0),
        G is A/B.
division(division(B,A),rest(C,D,E)) :-
        problem(numbers(N1,N2,N3,N4,N5),goal(G)),
        combos(set(N1,N2,N3,N4,N5),combo(A,B),extra(C,D,E)),
        not(A=0),
        G is B/A.

other_numbers(special(N),others(N2,N3,N4,N5)) :-
```

```
        problem(numbers(N,N2,N3,N4,N5),goal(_)).
other_numbers(special(N),others(N1,N3,N4,N5)) :-
        problem(numbers(N1,N,N3,N4,N5),goal(_)).
other_numbers(special(N),others(N1,N2,N4,N5)) :-
        problem(numbers(N1,N2,N,N4,N5),goal(_)).
other_numbers(special(N),others(N1,N2,N3,N5)) :-
        problem(numbers(N1,N2,N3,N,N5),goal(_)).
other_numbers(special(N),others(N1,N2,N3,N4)) :-
        problem(numbers(N1,N2,N3,N4,N),goal(_)).

rule(1,situation1,action1).
rule(2,situation2,action2).
rule(3,situation3,action3).
rule(4,situation4,action4).
rule(5,situation5,action5).
rule(6,situation6,action6).
rule(7,situation7,action7).
rule(8,situation8,action8).

solve(numbers(N1,N2,N3,N4,N5),goal(G)) :-
        establishCryptoProblem(numbers(N1,N2,N3,N4,N5),goal(G)),
        displayProblem,
        solveProblemHeuristically,
        displaySolution.
solveProblemHeuristically :-
        rule(Number, Situation, Action),
        write('considering rule '),write(Number),write(' ...'),nl,
        Situation,
        write('application of rule '),write(Number),write(' produces '),
        Action.
        solveProblemHeuristically.

% Heuristics
situation1 :-
        problem(Numbers,Goal),
        Goal = goal(0),
        Numbers = numbers(N1,N2,N3,N4,N5),
        member(0, [N1,N2,N3,N4,N5]).
action1 :-
        problem(Numbers,_),
        Numbers = numbers(N1,N2,N3,N4,N5),
        assert(solution(ex(N1, *,ex(N2, *,ex(N3, *, ex(N4,*,N5)))))).
situation2 :-
        problem(numbers(N1,N2,N3,N4,N5),goal(G)),
        member(G,[N1,N2,N3,N4,N5]),
        member(0,[N1,N2,N3,N4,N5]),
        not(G=0).
action2 :-
        problem(_,goal(G)),
        other_numbers(special(G),others(A,B,C,D)),
        assert(solution(ex(G,+,ex(A,*,ex(B,*,ex(C,*,D)))))).
situation3:-
        problem(_,goal(0)),
        doubleton.
action3:-
        doubleton(doubleton(A,B),rest(C,D,E)),
        assert(solution(ex(ex(A,-,B),*,ex(C,*,ex(D,*,E))))).
situation4:-
        addition(addition(_,_),rest(C,D,E)),
        member(0,[C,D,E]).
action4:-
        addition(addition(A,B),rest(C,D,E)),
        assert(solution(ex(ex(A,+,B),+,ex(C,*,ex(D,*,E))))).
```

```
situation5:-
        subtraction(subtraction(_,_),rest(C,D,E)),
        member(0,[C,D,E]).
action5:-
        subtraction(subtraction(A,B),rest(C,D,E)),
        assert(solution(ex(ex(A,-,B),+,ex(C,*,ex(D,*,E))))).
situation6:-
        multiplication(multiplication(_,_),rest(C,D,E)),
        member(0,[C,D,E]).
action6:-
        multiplication(multiplication(A,B),rest(C,D,E)),
        assert(solution(ex(ex(A,*,B),+,ex(C,*,ex(D,*,E))))).
situation7:-
        division(division(_,_),rest(C,D,E)),
        member(0,[C,D,E]).
action7:-
        division(division(A,B),rest(C,D,E)),
        assert(solution(ex(ex(A,/,B),+,ex(C,*,ex(D,*,E))))).
situation8:-
        problem(numbers(G,G,G,G,G),goal(G)).
action8:-
        problem(_,goal(G)),
        assert(solution(ex(G,-,ex(ex(G,-,G),-,ex(G,-,G))))).
```