Heuristic Crypto Problem Solver

The heuristic problem solver is a Prolog program that allows users to solve crypto problems heuristically. The program can solve single crypto problems as well as multiple random crypto problems at once. It implements eight different heuristics where the program tries to apply every single heuristic to the given crypto problem. If the program succeeds at one of the heuristics, then it prints "application of the rule (number of the heuristic) produces" followed by the crypto problem solution. Else, if after trying every heuristic no solution was found then it stops, and it prints that it considered the rules. The following are all the rules or facts that it considers for a given crypto problem.

```
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).
```

Each rule is composed of a number, a situation, and an action. The number identifies the rule, so the user knows which rule was considered and if the rule applies to the crypto problem then it lets the user know that the "application of the rule (number of the rule) produces" the solution to the problem. The following predicate is where Prolog tries every single rule and displays some output to the user accordingly:

```
solveProblemHeuristically  :-
        rule(Number,Situation,Action),
        write('considering rule '),write(Number),write(' ...'),nl,
        Situation,
         write('application of rule '),write(Number),write(' produces '),
        Action.
solveProblemHeuristically.
```

The situation is basically the definition of the heuristic or the condition. If the situation is not met by the crypto problem then it proceeds to the next situation, else, if the criteria of the situation are met then it proceeds to the next stage which is called the “action”. The action basically solves the crypto problem by using the heuristic definition. The following code is an example of what the definition of situation and action look like in Prolog:

```
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)))))).
```