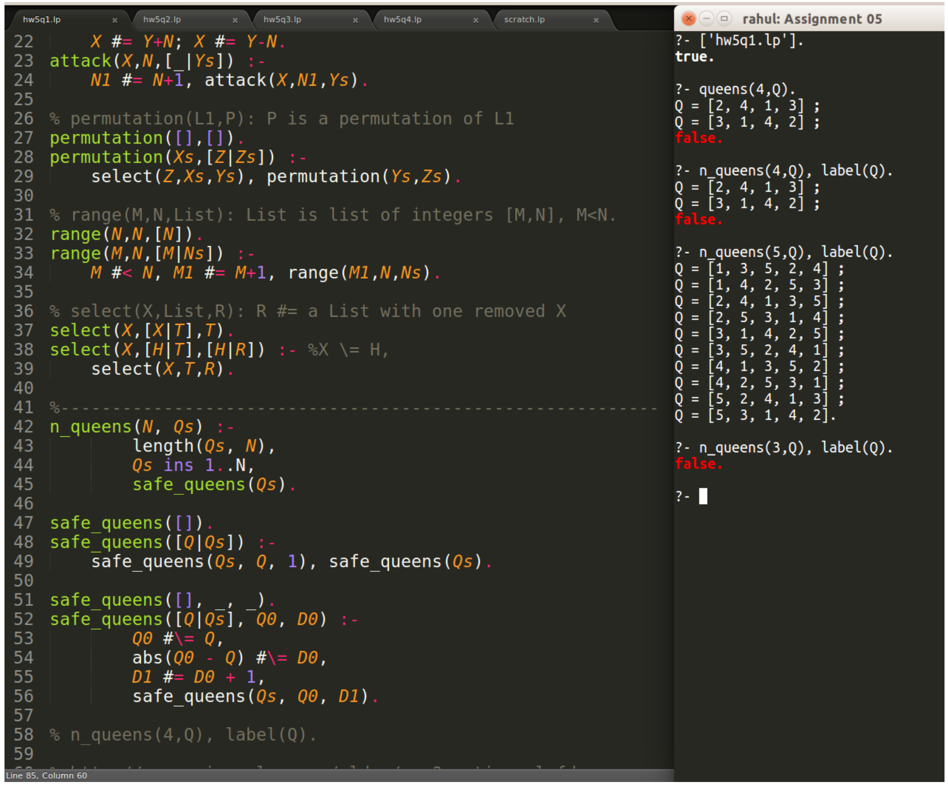
University of Texas at Dallas
CS 6374 - Computational Logic
Homework 05

Problem 1: Program the N-Queen problem in CLP(FD). Follow the structure of the 8 queens problem whose code is given to you.

Ans.

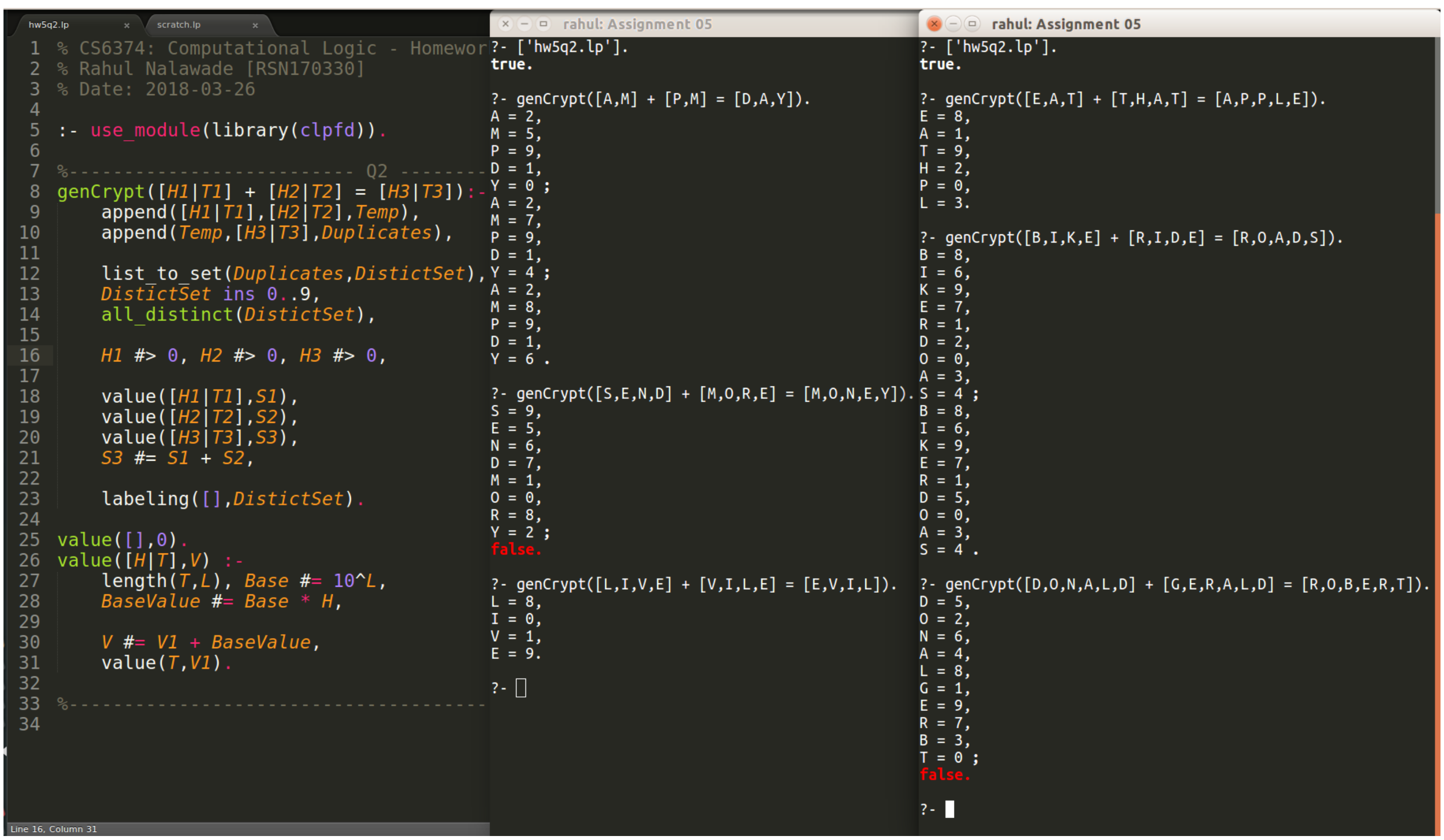
```
n_queens(N, Qs) :-
      length(Qs, N),
      Qs ins 1..N,
      safe_queens(Qs).

safe_queens([]).
safe_queens([Q|Qs]) :-
      safe_queens(Qs, Q, 1),
      safe_queens(Qs).

safe_queens([], _, _).
safe_queens([Q|Qs], Q0, D0) :-
      Q0 #\= Q,
      abs(Q0 - Q) #\= D0,
      D1 #= D0 + 1,
      safe_queens(Qs, Q0, D1).

% n_queens(4,Q), label(Q).
```



Problem 2: Write a CLP(FD) program to solve cryptarithmetic addition problems.

Ans.

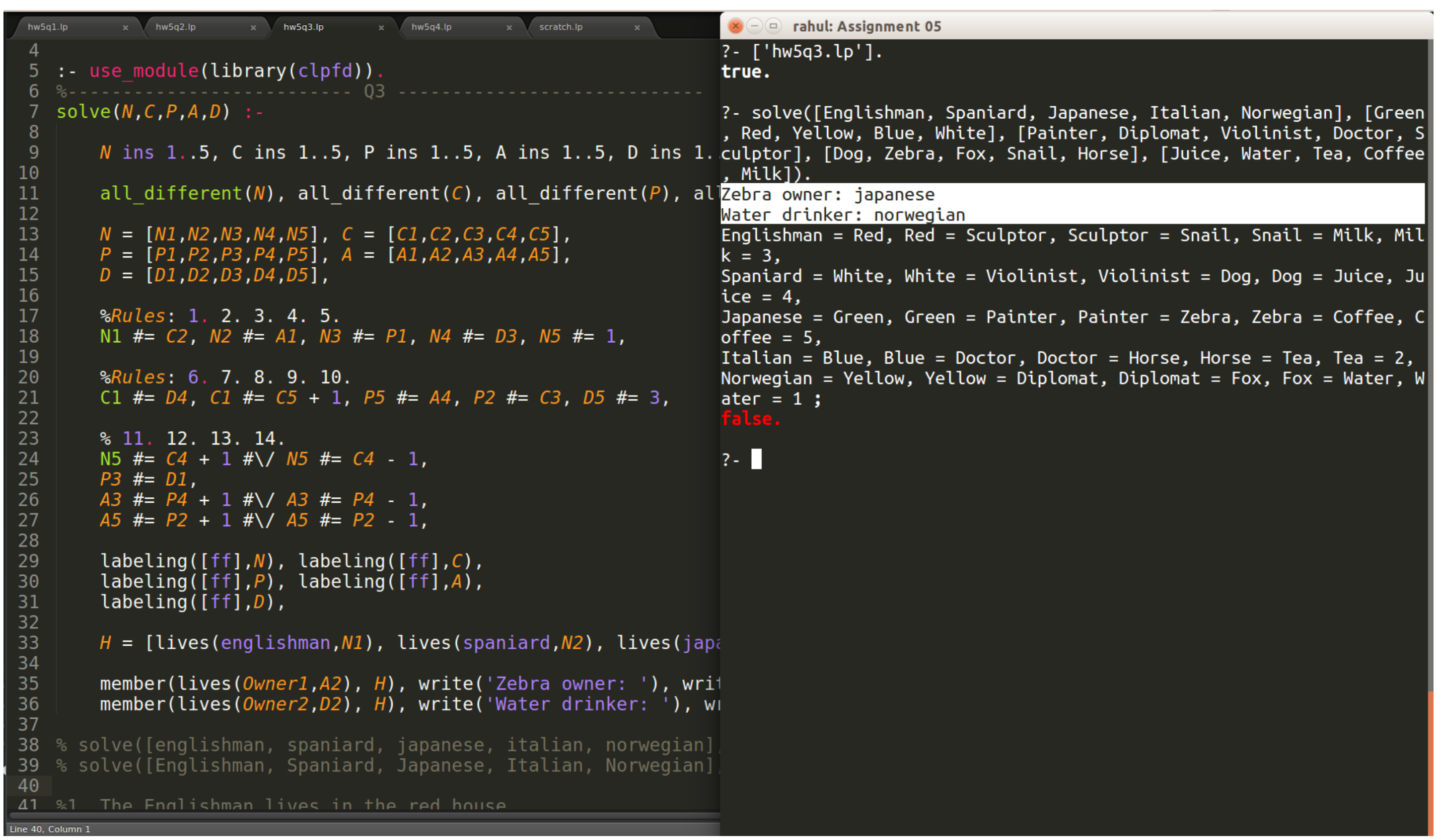
```
genCrypt([H1|T1] + [H2|T2] = [H3|T3]):-
	append([H1|T1],[H2|T2],Temp),
	append(Temp,[H3|T3],Duplicates),

	list_to_set(Duplicates,DistictSet),
	DistictSet ins 0..9,
	all_distinct(DistictSet),

	H1 #> 0, H2 #> 0, H3 #> 0,

	value([H1|T1],S1),
	value([H2|T2],S2),
	value([H3|T3],S3),
	S3 #= S1 + S2,

	labeling([],DistictSet).

value([],0).
value([H|T],V) :-
	length(T,L), Base #= 10^L,
	BaseValue #= Base * H,

	V #= V1 + BaseValue,
	value(T,V1).
```



Problem 3: Program the Zebra puzzle in CLP(FD).

Ans.

```
solve(N,C,P,A,D) :-
      N ins 1..5, C ins 1..5, P ins 1..5, A ins 1..5, D ins 1..5,
      all_different(N), all_different(C), all_different(P), all_different(A),
all_different(D),

      N = [N1,N2,N3,N4,N5], C = [C1,C2,C3,C4,C5], P = [P1,P2,P3,P4,P5],
      A = [A1,A2,A3,A4,A5], D = [D1,D2,D3,D4,D5],

      %Rules: 1. 2. 3. 4. 5.
      N1 #= C2, N2 #= A1, N3 #= P1, N4 #= D3, N5 #= 1,

      %Rules: 6. 7. 8. 9. 10.
      C1 #= D4, C1 #= C5 + 1, P5 #= A4, P2 #= C3, D5 #= 3,

      %Rules: 11. 12. 13. 14.
      N5 #= C4 + 1 #\/ N5 #= C4 - 1, P3 #= D1, A3 #= P4 + 1 #\/ A3 #= P4 - 1,
      A5 #= P2 + 1 #\/ A5 #= P2 - 1,

      labeling([ff],N), labeling([ff],C), labeling([ff],P), labeling([ff],A),
      labeling([ff],D),

      H = [lives(englishman,N1), lives(spaniard,N2), lives(japanese,N3),
lives(italian,N4), lives(norwegian,N5)],

      member(lives(Owner1,A2), H), write('Zebra owner: '), write(Owner1), nl,
      member(lives(Owner2,D2), H), write('Water drinker: '), write(Owner2), nl.
```

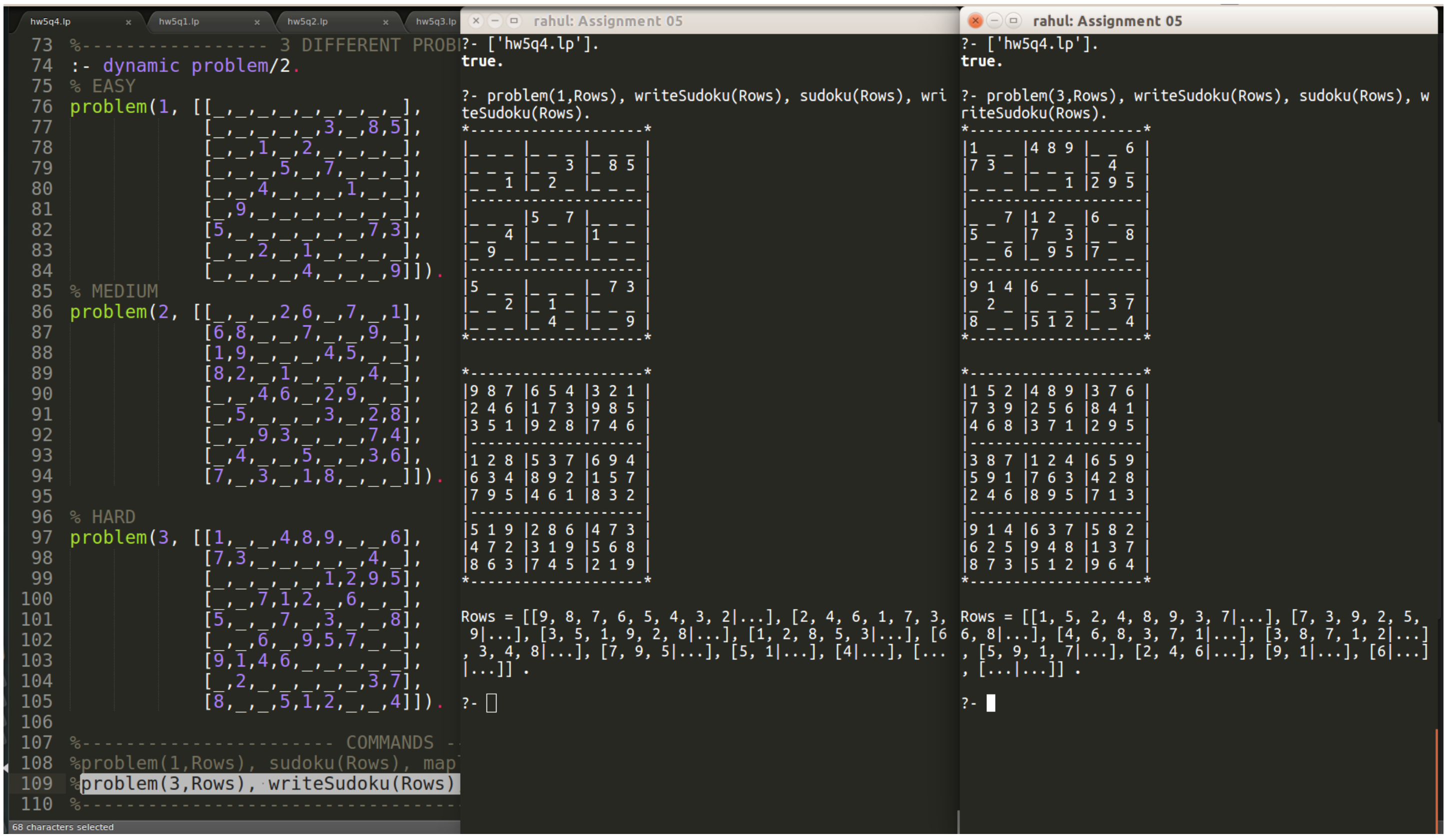


Problem 4: Program the Sudoku Puzzle in CLF(FD). You should read input from the user which consists of a series of terms of the form:
f(X,Y,Z).
in a file called "input". The term f(X,Y,Z) states that the square at position (X, Y) has value Z (Z ∈ 1..9). The input file is used to indicates the values given at the various squares in the puzzle. Your program should print the solution on the screen using write statements.

Ans.

```
sudoku(Rows) :-
      length(Rows, 9), maplist(same_length(Rows), Rows),
      append(Rows, Vs), Vs ins 1..9,
      maplist(all_distinct, Rows),
      transpose(Rows, Columns),
      maplist(all_distinct, Columns),
      Rows = [A,B,C,D,E,F,G,H,I],
      blocks(A, B, C), blocks(D, E, F), blocks(G, H, I).

blocks([], [], []).

blocks([A,B,C|Bs1], [D,E,F|Bs2], [G,H,I|Bs3]) :-
      all_distinct([A,B,C,D,E,F,G,H,I]),
      blocks(Bs1, Bs2, Bs3).

%----------------------- PRINTING ------------------------

writeRow([H1,H2,H3,H4,H5,H6,H7,H8,H9]) :-
      write('|'),
      %(var(H1) -> write('_'); write(H1)), write(' '),
      (var(H1) -> write('_'); write(H1)), write(' '),
      (var(H2) -> write('_'); write(H2)), write(' '),
      (var(H3) -> write('_'); write(H3)), write(' '), write('|'),
      (var(H4) -> write('_'); write(H4)), write(' '),
      (var(H5) -> write('_'); write(H5)), write(' '),
      (var(H6) -> write('_'); write(H6)), write(' '), write('|'),
      (var(H7) -> write('_'); write(H7)), write(' '),
      (var(H8) -> write('_'); write(H8)), write(' '),
      (var(H9) -> write('_'); write(H9)), write(' '),
      write('|'), nl.

%writeSudoku([]) :- nl.
writeSudoku([H1,H2,H3,H4,H5,H6,H7,H8,H9]) :-
      writeOuterLine([1,2,3,4,5,6,7,8,9]),
      writeRow(H1),
      writeRow(H2),
      writeRow(H3), writeInnerLine([1,2,3,4,5,6,7,8,9]),
      writeRow(H4),
      writeRow(H5),
      writeRow(H6), writeInnerLine([1,2,3,4,5,6,7,8,9]),
      writeRow(H7),
      writeRow(H8),
      writeRow(H9),
      writeOuterLine([1,2,3,4,5,6,7,8,9]), nl.
```

```
writeInnerLine([H|T]) :-
      write('|'),
      length([H|T],L), Temp is sqrt(L), N is ceil(Temp),
      Dashes is N+L*2-1, writeInnerDashes(Dashes).

writeInnerDashes(0) :- write('|'), nl.
writeInnerDashes(N) :- write('-'), N1 is N - 1, writeInnerDashes(N1).

writeOuterLine([H|T]) :-
      write('*'),
      length([H|T],L), Temp is sqrt(L), N is ceil(Temp),
      Dashes is N+L*2-1, writeOuterDashes(Dashes).

writeOuterDashes(0) :- write('*'), nl.
writeOuterDashes(N) :- write('-'), N1 is N - 1, writeOuterDashes(N1).

%----------------- 3 DIFFERENT PROBLEMS -------------------
:- dynamic problem/2.
% EASY
problem(1, [[_,_,_,_,_,_,_,_,_],
                  [_,_,_,_,_,3,_,8,5],
                  [_,_,1,_,2,_,_,_,_],
                  [_,_,_,5,_,7,_,_,_],
                  [_,_,4,_,_,_,1,_,_],
                  [_,9,_,_,_,_,_,_,_],
                  [5,_,_,_,_,_,_,7,3],
                  [_,_,2,_,1,_,_,_,_],
                  [_,_,_,_,4,_,_,_,9]]).
% MEDIUM
problem(2, [[_,_,_,2,6,_,7,_,1],
                  [6,8,_,_,7,_,_,9,_],
                  [1,9,_,_,_,4,5,_,_],
                  [8,2,_,1,_,_,_,4,_],
                  [_,_,4,6,_,2,9,_,_],
                  [_,5,_,_,_,3,_,2,8],
                  [_,_,9,3,_,_,_,7,4],
                  [_,4,_,_,5,_,_,3,6],
                  [7,_,3,_,1,8,_,_,_]]).

% HARD
problem(3, [[1,_,_,4,8,9,_,_,6],
                  [7,3,_,_,_,_,_,4,_],
                  [_,_,_,_,_,1,2,9,5],
                  [_,_,7,1,2,_,6,_,_],
                  [5,_,_,7,_,3,_,_,8],
                  [_,_,6,_,9,5,7,_,_],
                  [9,1,4,6,_,_,_,_,_],
                  [_,2,_,_,_,_,_,3,7],
                  [8,_,_,5,1,2,_,_,4]]).
```



Note:

* All **source files** for each problem **'i'** is saved as **hw5q*i*.lp**. Please, refer them for evaluation.

* All the **screen shots** in this document are also attached as **a5q*i*.png**.