Assignment - 4

Sarvansh Prasher

Prolog Code :-

```
% Assignment 4
% @author - Sarvansh Prasher
% @version - 1.0

%----------%%----------%%----------%%----------%%----------%%----------
% Question 1
% Test query "queens(8,Qs)"

:- use_module(library(clpfd)).
:- use_rendering(chess).
queens(N, Qs) :- length(Qs, N), Qs ins 1..N,all_distinct(Qs),
                    queen_position(Qs),labeling([ffc],Qs).


queen_position([]).

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

queen_position([],_,_).

queen_position([Q|Qs],Qr,R) :- abs( Qr- Q) #\= R,
                                        R1 #= R + 1, queen_position(Qs,Qr,R1).


%----------%%----------%%----------%%----------%%----------%%----------

% Question 2
% Test query : problem(1, Rows), sudoku(Rows).
:- use_module(library(clpfd)).
:- use_rendering(sudoku).

sudoku(Rows):- length(Rows,9), sudokuMain(Rows),transpose(Rows,Cols),sudokuMain(Cols),
                    Rows = [A,B,C,D,E,F,G,H,I],
                    sudokoBlock(A,B,C),sudokoBlock(D,E,F),sudokoBlock(G,H,I),
                    flatten(Rows,List),labeling([ffc],List).

sudokoBlock([],[],[]).
sudokoBlock([Block11,Block12,Block13|Block1],[Block21,Block22,Block23|Block2],
   [Block31,Block32,Block33|Block3]) :-

all_distinct([Block11,Block12,Block13,Block21,Block22,Block23,Block31,Block32,Block33]),
        sudokoBlock(Block1,Block2,Block3).

sudokuMain([]).
sudokuMain([Rows|RowsQ]):- length(Rows,9), Rows ins
1..9,all_distinct(Rows),sudokuMain(RowsQ).


problem(1, [[_,_,6, 5,9,_, _,_,_],
            [_,_,3, _,_,_, _,7,_],
            [_,_,_, _,_,_, 5,6,_],

            [_,2,_, 1,7,_, _,_,_],
            [4,8,5, _,_,_, _,_,_],
            [_,6,_, _,_,4, 9,_,_],

            [2,_,_, _,_,5, _,_,8],
            [_,3,8, _,_,1, _,_,_],
            [_,_,_, 3,_,_, 7,5,4]]).

%----------%%----------%%----------%%----------%%----------%%----------

% Question 3
% Test query : color_map(L).

:- use_module(library(clpfd)).

color_map(L) :- vertices(V),length(V,Vertices),length(Colors,Vertices),
        Colors ins 1..4, mapNeighbors(V,V,Colors),
        finalMap(V,Colors,[],List), reverse(List,L),label(Colors).

mapNeighbors([],_,_).
mapNeighbors([Head|Tail], V, C):-edge(Head, List),colorSelect(Head, V, C, HeadColor),
        mapConstraint(HeadColor, List, V, C),mapNeighbors(Tail, V, C).

finalMap([],[], L, L).
finalMap([Head|Tail], [ColorH|ColorT], L, Result):- color(ColorH, C),
    Element = [Head, C], finalMap(Tail,ColorT,[Element|L], Result).

mapConstraint(_,[],_,_).
mapConstraint(Color, [Head|Tail], V, C):-colorSelect(Head,V,C, HeadColor),
        Color #\= HeadColor,mapConstraint(Color, Tail, V, C).

colorSelect(Color, [Color|_], [Head|_], Head).
colorSelect(Color, [Head|Tail], [_|T], List):-Color #\= Head,
        colorSelect(Color, Tail, T, List).

color(1,green).
color(2,red).
color(3,yellow).
color(4,blue).

edge(1,[2,3,4,6]).
edge(2,[1,3,5]).
edge(3,[1,2,4,5,6]).
edge(4,[1,3,5,6]).
edge(5,[2,3,4]).
edge(6,[1,3,4]).

vertices([1,2,3,4,5,6]).


%----------%%----------%%----------%%----------%%----------%%----------

% Question 4
% Test query : solveZebra(Zebra,Water).

:- use_module(library(clpfd)).

solveZebra(Zebra,Water):-

Nationality = [English,Spanish,Ukrainian,Norwegian,Japanese],
Drinks =    [Coffee,Water,Milk,Juice,Tea],
Colors =    [Red,Green,Blue,White,Yellow],
Animals =   [Zebra,Horse,Dog,Serpent,Fox],
Cigratters = [LuckyStrike,Winston,_Chesterfields,Kool,Kent],

Nationality ins 1..5,
Drinks ins 1..5,
Colors ins 1..5,
Animals ins 1..5,
Cigratters ins 1..5,
English #= Red,
Spanish #= Dog,
Coffee #= Green,
Ukrainian #= Tea,
Green #= White-1 #\/ Green #= White+1,
Winston #= Serpent,
Yellow #= Kool,
Milk #= 3,
Norwegian #= 1,
Chesterfield #= Fox-1 #\/ Chesterfield #= Fox+1,
Kool#=Horse-1 #\/ Kool#=Horse+1,
LuckyStrike #= Juice,
Japanese #= Kent,
Norwegian #= Blue -1 #\/ Norwegian#=Blue+1,

all_distinct(Nationality),
all_distinct(Colors),
all_distinct(Drinks),
all_distinct(Animals),
all_distinct(Cigratters),

label(Nationality),
label(Drinks),
label(Colors),
label(Cigratters),
label(Animals).
```

Sample Run:

1. **?- queens(8,Qs)**

```
Qs = [1, 5, 8, 6, 3, 7, 2, 4]
Qs = [1, 6, 8, 3, 7, 4, 2, 5]
Qs = [1, 7, 4, 6, 8, 2, 5, 3]
Qs = [1, 7, 5, 8, 2, 4, 6, 3]
Qs = [2, 4, 6, 8, 3, 1, 7, 5]
….
```

2. **?-problem(1, Rows), sudoku(Rows).**

```
Rows = [[8, 1, 6, 5, 9, 7, 4, 3, 2], [9, 5, 3, 4, 2, 6, 8, 7, 1], [7, 4, 2, 8, 1, 3, 5, 6, 9], [3, 2, 9, 1, 7, 8, 6, 4, 5], [4, 8, 5, 6, 3, 9, 1, 2, 7], [1, 6, 7, 2, 5, 4, 9, 8, 3], [2, 7, 4, 9, 6, 5, 3, 1, 8], [5, 3, 8, 7, 4, 1, 2, 9, 6], [6, 9, 1, 3, 8, 2, 7, 5, 4]]
```

3. **color_map(L).**

```
L = [[1, green], [2, red], [3, yellow], [4, red], [5, green], [6, blue]]
L = [[1, green], [2, red], [3, yellow], [4, red], [5, blue], [6, blue]]
L = [[1, green], [2, red], [3, yellow], [4, blue], [5, green], [6, red]]
L = [[1, green], [2, red], [3, blue], [4, red], [5, green], [6, yellow]]
….
```

4. **solveZebra(Zebra,Water).**

```
Water = Zebra, Zebra = 1
Water = 1,
Zebra = 5
Water = Zebra, Zebra = 1
Water = 1,
Zebra = 4
```