CSPC 54 : Prolog Assignment-9

Name : Rajneesh Pandey,

Roll no. 106119100,

Class : CSE-B

## Instructions 1

**A picture containing shape

Description automatically generated**

**CODE :**

:- use\_rendering(sudoku).

:- use\_module(library(clpfd)).

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 = [As,Bs,Cs,Ds,Es,Fs,Gs,Hs,Is],

        blocks(As, Bs, Cs),

        blocks(Ds, Es, Fs),

        blocks(Gs, Hs, Is).

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

blocks([N1,N2,N3|Ns1], [N4,N5,N6|Ns2], [N7,N8,N9|Ns3]) :-

        all\_distinct([N1,N2,N3,N4,N5,N6,N7,N8,N9]),

        blocks(Ns1, Ns2, Ns3).

problem(1, [[\_,\_,\_,\_,\_,\_,\_,\_,\_],

            [\_,\_,\_,\_,\_,3,\_,8,5],

            [\_,\_,1,\_,2,\_,\_,\_,\_],

            [\_,\_,\_,5,\_,7,\_,\_,\_],

            [\_,\_,4,\_,\_,\_,1,\_,\_],

            [\_,9,\_,\_,\_,\_,\_,\_,\_],

            [5,\_,\_,\_,\_,\_,\_,7,3],

            [\_,\_,2,\_,1,\_,\_,\_,\_],

            [\_,\_,\_,\_,4,\_,\_,\_,9]]).

problem(2, [[\_,4,\_, \_,\_,\_, 3,\_,\_],

            [2,5,7, 4,3,6, \_,\_,\_],

            [\_,\_,\_, 9,1,\_, \_,\_,\_],

            [7,6,\_, \_,4,\_, 2,\_,8],

            [\_,\_,3, \_,\_,\_, 7,\_,\_],

            [1,\_,4, \_,2,\_, \_,3,9],

            [\_,\_,\_, \_,8,1, \_,\_,\_],

            [\_,\_,\_, 6,5,7, 9,4,2],

            [\_,\_,5, \_,\_,\_, \_,8,\_]]).

**INPUT / OUTPUT:**

I/p =>

?- problem(1, Solution), sudoku(Solution), maplist(writeln, Solution).

O/p =>

Calendar

Description automatically generated

I/p =>

?- problem(2, Solution), sudoku(Solution), maplist(writeln, Solution).

O/p =>

Table, calendar

Description automatically generated

**Screenshot:**

**Calendar

Description automatically generated with medium confidence**

## Instructions 2 :

## 

**Code:**

:- use\_rendering(sudoSolver).

:- use\_module(library(clpfd)).

/\*solver receives list X, verifies that X is length 256\*/

/\*remaps hex char atomics to numeral atomics, solves and returns string R with answer\*/

solver(X,R):-

  X = [

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,

  \_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_

  ],

  remap(Y,X), /\*create Y with hex vals\*/

  sudoSolver(Y), /\*solve sudoku puzzle Y\*/

  remap(Y,X),  /\*X now has no open variables, close open variables in Y\*/

  atomic\_list\_concat(X,R).  /\*unify R with string for solution\*/

/\*Open variables should trivially unify\*/

num\_map(X,Y):-var(Y),var(X).

/\*Otherwise remap between valid decimals and hex chars\*/

num\_map(0,0).

num\_map(1,1).

num\_map(2,2).

num\_map(3,3).

num\_map(4,4).

num\_map(5,5).

num\_map(6,6).

num\_map(7,7).

num\_map(8,8).

num\_map(9,9).

num\_map(10,a).

num\_map(11,b).

num\_map(12,c).

num\_map(13,d).

num\_map(14,e).

num\_map(15,f).

/\*remap between hex char and numeral atomic list\*/

remap([],[]).

remap([H1|T1],[H2|T2]):-

  num\_map(H1,H2),

  remap(T1,T2).

/\*\*to matrix receives list X and makes matrix Y\*/

to\_matrix([],[]).

to\_matrix(X,Y):-

  X = [A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P|TailX],

  Y = [[A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P]|TailY],

  to\_matrix(TailX, TailY).

/\*creates matrix from list and solves sudoku\*/

sudoSolver(X):-

  to\_matrix(X,Y),

  Y = [A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P],

  sudoku(Y).

/\*solves sudoku puzzle for input 16x16 matrix holding values between 0-15\*/

sudoku(Row) :-

  append(Row, Vars),

  Vars ins 0..15,

  maplist(all\_distinct, Row),

  transpose(Row, Col),

  maplist(all\_distinct, Col),

  Row = [A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P],

  blocks(A,B,C,D),

  blocks(E,F,G,H),

  blocks(I,J,K,L),

  blocks(M,N,O,P),

  maplist(label, Row).

/\*checks each block in matrix is unique\*/

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

blocks([A,B,C,D|BL1], [E,F,G,H|BL2], [I,J,K,L|BL3], [M,N,O,P|BL4]) :-

  all\_distinct([A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P]),

  blocks(BL1, BL2, BL3, BL4).

**Input / Output :**

?- solver(X,Y).

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f],

[4, 5, 6, 7, 0, 1, 2, 3, c, d, e, f, 8, 9, a, b],

[8, 9, a, b, c, d, e, f, 0, 1, 2, 3, 4, 5, 6, 7],

[c, d, e, f, 8, 9, a, b, 4, 5, 6, 7, 0, 1, 2, 3],

[1, 0, 3, 2, 5, 4, 7, 6, 9, 8, b, a, d, c, f, e],

[5, 4, 7, 6, 1, 0, 3, 2, d, c, f, e, 9, 8, b, a],

[9, 8, b, a, d, c, f, e, 1, 0, 3, 2, 5, 4, 7, 6],

[d, c, f, e, 9, 8, b, a, 5, 4, 7, 6, 1, 0, 3, 2],

[2, 3, 0, 1, 6, 7, 4, 5, a, b, 8, 9, e, f, c, d],

[6, 7, 4, 5, 2, 3, 0, 1, e, f, c, d, a, b, 8, 9],

[a, b, 8, 9, e, f, c, d, 2, 3, 0, 1, 6, 7, 4, 5],

[e, f, c, d, a, b, 8, 9, 6, 7, 4, 5, 2, 3, 0, 1],

[3, 2, 1, 0, 7, 6, 5, 4, b, a, 9, 8, f, e, d, c],

[7, 6, 5, 4, 3, 2, 1, 0, f, e, d, c, b, a, 9, 8],

[b, a, 9, 8, f, e, d, c, 3, 2, 1, 0, 7, 6, 5, 4],

[f, e, d, c, b, a, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0],

Text

Description automatically generated with medium confidence

**Screenshot :**

**Graphical user interface, application

Description automatically generated**