Q: Mine Placement

Consider a rectangular m x n grid of square/cells. A square can have up to 8 neighbouring squares (moving one step horizontally- left/right, vertically- above/below- or diagonally).

Some squares have a number (between 1-8) written on them. Other squares are blank. You have to place mines in some of the blank squares so that every square with a number (say n) has EXACTLY n neighbouring cells with mines in them. A square that is numbered cannot have a mine.

```
Sample Code
%%% Input file for the above example - save as example1.lp. has 12 squares
numbered.
marked(1,2,3). % square 1, 2  (bottom row, second col.) is marked with 3
marked(1,4,3).
marked(2,6,3).
marked(3,1,1).
marked(3,3,3).
marked(3,4,4).
marked(4,3,2).
marked(4,4,4).
marked(4,6,3).
marked(5,1,2).
marked(6,3,2).
marked(6,5,3).
%%
%%%%%%%%%% template code file for version1 - mine1.lp
% Grid has r rows and c columns
rows(1..r).
cols(1..c).
% Each square has maximum of 8 neighbours.
nbr(0..8).
% The data file contains all the filled squares
% marked(I,J,N) means that square(I,J)
% is marked as having  N neighbours with mines.
% Use hasMine(I,J) to mean there is a mine in square(I,J).
% There is no mine in squares marked with a number.
%%%%% fill your code here...
#show hasMine/2.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sample run for above example...

clingo -c r=6 -c c=6 example1.lp mine1.lp

clingo version 5.1.0

Reading from mine3.lp ...

Solving...

Answer: 1

hasMine(1,1) hasMine(2,2) hasMine(2,3) hasMine(2,4) hasMine(1,5) hasMine(1,6) hasMine(3,5) hasMine(4,5) hasMine(5,2) hasMine(5,4) hasMine(5,5) hasMine(6,1) hasMine(6,6)

SATISFIABLE

%%% Solution has placed 13 mines in empty squares. Verify with the answer shown in the image above... This answer is also unique. You can try with clingo 0 ....

Q2: Mine Placement (Liar Version)

Input is the same- some squares are numbered from 1 to 8. Only this time, every number is off by one. That is, a square numbered 3 does not have exactly 3 neighbouring squares with mines. It has either 2 neighbours with mines OR 4 neighbours with mines. That is, all numbers are wrong. Some have underestimated by 1, some have overestimated by 1. Your job is to place the MINIMUM number of mines in some empty squares so that all the numbered cells are wrong by 1 (more by 1 or less by 1).

```
%% sample input example1.lp is same as above.
%% You can construct more simple examples - just number one of the central
squares with 3 say.
%% template file for mine2.lp code is the same as above.
%% sample output - please draw on a 6x6 grid in your notebook and verify...
clingo -c r=6 -c c=6   example1.lp mine2.lp
clingo version 5.1.0
Reading from example1.lp ...
Solving...
Answer: 1
hasMine(1,1) hasMine(2,1) hasMine(2,4) hasMine(2,5) hasMine(3,6) hasMine(3,2)
hasMine(4,5) hasMine(5,3) hasMine(5,4) hasMine(6,2) hasMine(6,6)
Optimization: 11
Answer: 2
hasMine(1,1) hasMine(2,1) hasMine(2,4) hasMine(2,5) hasMine(3,6) hasMine(4,2)
hasMine(4,5) hasMine(5,3) hasMine(5,4) hasMine(6,4)
Optimization: 10
Answer: 3
hasMine(1,1) hasMine(2,2) hasMine(2,4) hasMine(2,5) hasMine(1,6) hasMine(4,1)
hasMine(4,5) hasMine(5,4) hasMine(5,5)
Optimization: 9
Answer: 4
hasMine(2,2) hasMine(2,3) hasMine(2,5) hasMine(1,6) hasMine(4,1) hasMine(4,5)
hasMine(5,4) hasMine(5,5)
Optimization: 8
Answer: 5
hasMine(2,2) hasMine(2,3) hasMine(2,5) hasMine(3,5) hasMine(4,1) hasMine(5,4)
hasMine(5,5)
Optimization: 7
OPTIMUM FOUND
Models      : 5
Optimum     : yes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```