FIT5216: Modelling Discrete Optimization Problems
Inclass Task 9: Hidato

1 Problem Statement

Hidato is a puzzle devised by Dr. Gyora M. Benedek, an Israeli mathematician. The aim is to fill a grid with consecutive numbers that connect horizontally, vertically or diagonally.

The player is given a partially filled in $n \times m$ grid of and needs to fill the cells with numbers from 1 to nm.

The rules are that each number appears exactly once, and any adjacent numbers k and $k+1$ must appear adjacent in the grid, which means either horizontally, vertically or diagonally.

Consider the example Hidato puzzle for a 6×6 grid below

26	28	29	31	33	34
			30		
3	1	22		36	
4	2				18
	8		13	17	
7			11	14	15

Usually the first and last number, here 1 and 36, must be given in the clues. We will ignore this requirement.

The unique solution is below, notice how each number is adjacent to its neighbouring numbers.

26	28	29	31	33	34
27	25	24	30	32	35
3	1	22	23	36	19
4	2	9	21	20	18
5	8	10	13	17	16
7	6	12	11	14	15

The MiniZinc model should make use of parameter definitions

```
int: n; % no of ROWS
set of int: ROW = 1..n;
int: m; % no of COLS
set of int: COL = 1..m;

set of int: CLUE = 0..m*n;
array[ROW,COL] of CLUE: clue;
```

The `clue` values are either a number in $1..nm$ which means that that number must appear in that grid cell, or 0 meaning the grid cell is currently unassigned. For example, the data for the example above is given as

```
n = 6;
m = 6;

clue = [| 26,28,29,31,33,34
        |  0, 0, 0,30, 0, 0
        |  3, 1,22, 0,36, 0
        |  4, 2, 0, 0, 0,18
        |  0, 8, 0,13,17, 0
        |  7, 0, 0,11,14,15
        |];
```

Build a MiniZinc model `hidato.mzn` to find a solution. Use the variable declarations and output:

```
array[ROW,COL] of var NUM: x;
output ["x = array2d(ROW,COL,\(x));\n"];
```

Hint: Hidato problems are *very easy* for CP with the correct model. Remember the material we covered this week, and try to use it.

2 Instructions

Edit the provided `mzn` model files to solve the problems described above. Your implementations can be tested locally by using the *Run* icon in the MiniZinc IDE or by using,

```
minizinc ./modelname.mzn ./datafile.dzn
```

at the command line.