Game of Life

Purpose

The goal of this task is to let us see how you approach a problem. There are numerous valid ways to implement this task, we simply like to see how you would do it.

Note that

- You **don't need** to implement a UI for the game.
- We estimate this at no more than 2-3 hours. Don't waste a day on this.
- Build it with the same quality you'd like to see in your production code. Consider tests, conventions, comments and documentation.

Requirements

We, at Panoply.io, decided to launch the next generation of Conway's Game of Life!

The original universe of the Game of Life is a two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead. Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent.

At each step in time, the following transitions occur:

1. Any live cell with fewer than two live neighbors dies (under population).
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies (overpopulation).
4. Any dead cell with exactly three live neighbors becomes a live cell (reproduction).

The grid is initiated with a seed - an initial state of alive and dead cells, and continues on to its next generation automatically. This process repeats itself indefinitely, or until a predefined number of generations pass.

In the new incarnation Game of Life: Infection, we added a new phase. After a predefined number of generations has reached, a virus is spread around the universe infecting cells.

From now on the rules change, the new rules are:

1. Any dead cell with a single live neighbor lives on to the next generation.
2. Any live cell with no horizontal or vertical live neighbors dies.

Input

The program should accept the following arguments:

Argument	Type	Description
width	int	The width of the world
height	int	The height of the world
infect-after	int	The number of generations after which the infection stage will start
max-generations	int	The maximum number of generations that can be created. Including all phases of the game
seed	[] int	The initial state of the world

Example

A *suggested* way to pass the parameters is:

program [width] [height] [infect-after] [max-generations] [seed]

```
$ your-program 2 3 3 6 "1 0 0 1 1 1"
```

Output

Each generation should produce a flat output of the current state to **stdout.** The values should be separated by whitespace and followed by a newline character at the end.

The format for the output is: **<row 1> <row 2> <row 3> ... <row n>\n**

Example

With the current state being

```
0 0 0
1 0 0
1 0 1
```

the **first 2** lines your program should output to stdout will be

```
0 0 0 1 0 0 1 0 1
0 0 0 0 1 0 0 1 0
... more lines to come
```