Game of Life

This is a **challenge** exercise. It is **not compulsory**, and may be completed **individually or with your lab partner**.

For this activity, you will need to complete a program called `life.c` .

Make sure you have completed cellular before attempting this activity.

Download `life.c`, or copy it into your current directory on a CSE system by running

```
$ cp /web/cs1511/17s2/week05/files/life.c .
```

This program is a simple life simulator. It is an example of a 2-dimensional cellular automaton, similar to Conway's Game of Life. In this simulation, the world is represented by *two-dimensional* array. Each element of the array is a single *cell* which can contain up to 1 creature. If it contains a creature, we call it `ALIVE` . If it contains no creatures, we call it `DEAD` .

The simulation will advance between *generations*. Every time it moves to a new *generation* each cell may

- become `ALIVE` ,
- become `DEAD` ,
- stay `ALIVE` , or
- stay `DEAD` .

Your task is to complete the function `nextGeneration` , which takes in an array of the current *generation* and the next *generation*, and sets all of the values of the cells in the next *generation*. Currently, this function makes every generation the same.

This version of the simulator is more complex than cellular. In this version, each cell can have neighbours in two directions, left and right (which are on the same array), and above and below (which are on the previous and next arrays). This version also has *diagonal* neighbours.

The rules for what happens on each *generation* are:

- If a cell in the current generation is `DEAD` , then in the next generation
    - it becomes `ALIVE` if it has exactly three `ALIVE` neighbours, otherwise
    - it remains `DEAD` .
- If a cell in the current generation is `ALIVE` , then in the next generation
    - it remains `ALIVE` if it has two or three `ALIVE` neighbours, otherwise
    - it becomes `DEAD` .
- The neighbours of a cell are the cells that are
    - next to it, in its own array,
    - above it, in the previous array,
    - below it, in the next array,
    - diagonal to it, next to the cells in the previous or next array.
- The edges, before the first row and column and after the last row and column, are considered `DEAD`

There are no tests for this activity, so feel free to have some fun. Try and see what happens to the output if you change the rules.

To run Styl-o-matic:

```
$ 1511 stylomatic life.c
Looks good!
```

You'll get advice if you need to make changes to your code.

Submit your work with the *give* command, like so:

```
$ give cs1511 wk05_life
```

Or, if you are working from home, upload the relevant file(s) to the `wk05_life` activity on Give Online.