We expect that you will complete mini projects ***individually***. This means that you ***cannot***:

- Share code with friends.
- Find code on the internet.
- Write code together with a partner.
- Submit ***any*** code that was not written entirely by you.

This does leave room for limited collaboration. The following things ***are acceptable:***

- Talking to a partner about strategies you are using to code this project.
- Asking for help ***with code that you have written***.
- Offering to help a friend ***with code that they have written*** as long as you do not give them your code, or copy their code.
- Asking instructors for help.

Mini Project 3: Conway's Game of Life

For this project, you will write a program that will have two inputs:

- A 2D matrix of 1's and 0's.
- A number of iterations to perform Conway's Game of Life.

It should have one output:

- The final 2D matrix after `n` iterations of Conway's Game of Life.

Conway's Game of Life is a "0 player" computer game. It begins with a matrix where each pixel has one of two values: alive (1) and dead (0). It then updates that matrix **iteratively** (repeatedly, following the same set of rules) to determine which pixels are now alive and which pixels are now dead. Following a few, very simple rules, Conway's Game of Life creates a number of interesting, repeatable geometric patterns from seemingly random inputs. To get a good feel for what those patterns are, go to this website:

https://bitstorm.org/gameoflife/

Conway's Game of Life has the following rules for updating the matrix each iteration:

- Pixels with the value of 1 are alive.
- Pixels with the value of 0 are dead.
- Pixels that are alive during this iteration with 2-3 neighbors that are alive (not including themselves) stay alive for the next iteration.
- Pixels that are dead during this iteration with exactly 3 neighbors turn alive for the next iteration.
- All other pixels are dead in the next iteration.

These rules should be applied to update the image matrix for each iteration of the game, until n iterations have been performed.

You may find that the code that you created in the "Border of zeros" activity from Lecture 6 helpful in this project.

For example, the result of the following input `mat`:

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

For any even numbered `n`, your input will be the same as your output. For any odd numbered `n`, your output would be the following matrix:

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

To test your code you may wish to create a visualization of Conway's game. To do this, displaying your matrix as an image can be useful.

Once you have an iterative version of Conway's Game running, you can add the following code at the end of your iterating loop:

```
imagesc(mat)
pause(0.5)
```

To create random 10x10 inputs, use the following code:

```
randomMat = randn(10,10) > 0;
```

This will allow you to see each iteration of Conway's Game in action for random matrices. See if you can spot repeating patterns! If you are unsure whether your patterns are correct, try going to the web app implementation of Conway's game to look for similar patterns.