**MATLAB Bootcamp: GOL Project Week 2**

*Conway’s Game of Life*

***INSTRUCTIONS:***This assignment covers topics through module 8. Submit the solutions on Blackboard as described below. Use the following nomenclature for submitting the final solutions:

For problems 2a,2b: **GOL.m, yourlastname\_problem2b.png.**

**Make sure to comment each part of your code.**

For problem 2c: **yourlastname\_problem2c.png, RLE\_decoder.m, modified\_input\_pattern.txt**

**Add in any text/RLE files that you used in your code**

Problem 2a: Functionalize the Game of Life – *30 points*

Functions: Now that we have a working script, we can place the game of life (GOL) script into a function that accepts the binary input, and outputs the subsequent generation.

The function GOL should take the form:

function next\_gen\_mat = GOL(in\_mat)

The function should:

* Should find and assign the x/y dimensions to x\_size and y\_size.
* Compute and output the next-generation matrix as a binary matrix of the same size, with boundary conditions calculated in Week 1.

Submit part a as **GOL.m**

Problem 2b: Run 100 Generations – *20 points*

Now that you have the for-loop ready, run 100 generations of the [Pi-heptomino](https://www.conwaylife.com/wiki/Pi-heptomino) by calling the function and assigning it to the same variable within a loop. The input pattern should make sure that the initial matrix is correctly padded on all sides with zeros, so it has room to evolve. Plot the final shape and export as a png. The initial pattern will be considered the 0th generation.

Submit part b as **yourlastname\_problem2b.png.**

Problem 2c: Import RLEs – *50 points*

With a custom function to create the next generation and a way of simulating the game any amount of generations, the only thing left is to be able to import new initial patterns to simulate from the web! To do this, you need a better understanding of the format which these binary matrices are converted to for storage on the web:

<http://www.conwaylife.com/wiki/Run_Length_Encoded>

To import and convert these files into binary matrices that can be used in your script, you have to download and import the letters into MATLAB. For this, *importdata()* will be useful. Decoding the string into a matrix can be done using the RLE rules found in the link above. Find matches of characters and numbers is best done using MATLAB’s built-in *regexp()* (regular expression). Look into the documentation to find how to use it. Another hint: *strfind()* is a useful way to find exact locations for any letter/number combination inside the *regexp*() string.

Finally, incorporate the RLE\_decoder into a function, and combine it with the scripts from 2a and 2b to create a script that is able to take any RLE file from the lifewiki, and simulate a given number of simulations of the script.

Import the [Traffic Jam](https://www.conwaylife.com/wiki/Traffic_jam) pattern through the RLE decoder and output the pattern that occurs at generation 23.

Hints: Under the pattern files, you will see the RLE. Open it with a text editor so that you can remove any headers found within it to make it compatible with importdata().

If you are having trouble importing the correct input pattern, try a simpler one initially (with fewer live cells) to make debugging easier, such as the blink or beacon patterns from the previous assignment.

**Submit your assignment as yourlastname\_problem2c.png, RLE\_decoder.m, modified\_input\_pattern.txt**

**And add any other code/text files that you used.**

**Bonus (10 points):**

Find an interesting RLE on the life wiki and upload a movie of the plots generated. Explain why you chose that specific input pattern. Hint: getframe() and videowriter() are useful built-in functions.