ENGR 1120 – Programming for Engineers (MATLAB)

Laboratory #10 – The Game of Life

**Objectives**: Practice the creation of a user-defined function from scratch while using a flowchart for development. Demonstrate the use of a case statement within a set of nested for loops to apply a set of conditions to elements in a 2-d array (matrix).

The game of life is a 'cellular automaton' that was invented by Cambridge University mathematician John Conway. The game starts with a grid (2-d array) of cells that all begin the game as either a living (value of 1) or a non-living (value of 0) cell. The rules are then applied to determine, for each future generation, which cells are born (become living), remain living, remain non-living, or die (become non-living). Internal cells are evaluated and changed for each new generation. The border cells can be treated in many ways. For this program the border cells will be considered inert, in other words they will always remain non-living.

# Beginning the Game

Begin the game by loading the map from the file given ‘gol\_map1.txt’. The data from the map should be saved as a 2D array of ones and zeros.

# The Rules

The following rules are applied only to the internal cells of the grid since the boarders are inert or always remain non-living. Remember that life is represented by a value of 1 and a non-living cell will have a value of 0.

* A living cell with 4 or more live neighbors - dies due to over-population.
* A living cell with less than 2 live neighbors - dies due to under-population.
* A non-living cell with exactly 3 live neighbors - is born
* All other cells remain unchanged.

# Ending the Game

Some initial configurations will reach a point of stability while others will seem to continue to change indefinitely. One case of stability is where there are no changes taking place from one generation to the next. Another case of stability will exist when two (or more) configurations oscillate, this will continue indefinitely.

What to do

First develop a flowchart for a user-defined function that will take as input the current generation's configuration of the game grid (a 2-d array of 1’s and 0’s) and will produce as output the next generation's configuration for the game grid. You will need to use a set of nested loops to evaluate only the internal cells of the game grid. Also, you cannot update (change the values) the one matrix as you go since doing so would change the evaluation of the cells being analyzed as the process continues. In other words, if you change the values in row 2 as you decide the fate of these cells, the new values will be accessed when determining the fate of the cells in row 3 as row 2 is a neighbor of row 3.

|  |  |  |
| --- | --- | --- |
| grid(i-1,j-1) | grid(i-1,j) | grid(i-1,j+1) |
| grid(i,j-1) | grid(i,j) | grid(i,j+1) |
| grid(i+1,j-1) | grid(i+1,j) | grid(i+1,j+1) |

Convert the flowchart into a MATLAB user-defined function and test the function thoroughly. Stay tuned for more instructions!