# COSC 251 – Programming Languages
# Project 4
# Spring 2019

**Objective:** Implement a classic "game" in LISP.

**Your Task:** You and your group will create a LISP application that simulates a variant of Conway's Game of Life. The Game of Life is played on an n x n board, where each cell of the board is a tile that has two states, either live or dead. Beginning in some initial configuration, a tile will either become alive, or die off, depending on the situation. These original rules are as follows:

- Any live cell with fewer than two live neighbors dies
- Any live cell with two or three live neighbors lives on
- Any live cell with four neighbors dies
- Any dead cell with exactly three live neighbors becomes alive

There are many variants of the Game of Life, and you will be exploring a way to model a generalization of the Game of Life. In this generalization, the rules for life or death are adjustable from call to call. For instance, you might allow for the rule for death to go from 0, 1, 2, and 4 neighbors to be just 0, and 1. Or for a dead cell to become alive when it is neighboring 1 or more live tiles. These will be parsed from input (we'd suggest storing them as a list). In addition, we will allow for a slight change of the definition of neighbors. Normally, we would only consider neighbors that are one step around the cell in the cardinal directions. For your simulation, we would also allow for longer distances. For instance, if the distance input is 3, we would consider any cell within 3 hops in any cardinal direction:

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | Live | | | |
| | | | | | | |
| | | Live | | | | |
| Live | | Live | ??? | | | Live |
| | | | | Live | | |
| | | | | | | |
| | | | Live | | | |

For this example the ??? cell has 5 live neighbors, and 7 dead neighbors.

For your simulation, you will create a simulation that allows for the following to be chosen upon execution:

- Size of the board (maximum 20 x 20).

- String representing the number of neighbors for a live cell to stay alive (for instance: “1 2 4” means that if the live cell is neighbored by 1, 2, or 4 live cells it stays alive).
- String representing the number of neighbors for a dead cell to become alive (see above for format example).
- Distance allowed for the neighbor rule.
- Number of live tiles to begin with (these will be randomly placed).

You should get all of this information from the user and output the board states one step at a time (starting with the initial, random placement), waiting for the user to hit the enter key before moving to the next step. An example execution output will be posted to the course website.

## Testing:

You should test your code using ReadyLisp or LispBox. Please specify in the comments which of the two interpreters you used.

Deliverables: the source file (.lisp) for your simulator.

**Expectations:** The code should be clean, concise, well-commented and correct. If you use an outside source, be sure to document that source. Significant use of outside sources will result in a deduction. You will work in teams of three for this project. Teams can be across sections, and you are allowed to choose your teams. If you choose your team, you must email one of the professors with the names of your team members by April 1st at 5pm. If you have not notified one of the professors by that point, you will be assigned a group.

DUE: April 17, 11:59pm via Blackboard, question cutoff 5pm, 4/17.

Potential Deductions:
-75 Does not compile
-50 Runs but crashes
-20 lack of comments, including header for every file (see project 1)
-30 no input allowed
-15 failure to take in input in the format specified (i.e. requiring each element of the live or death list to be input one at a time)
-10 failure to provide output in single steps
-20 does not handle variations in death or live rules
-15 does not handle distance variation
-40 does not output board states