# Seeking Alpha Engineering Test

Instructions:

This task should be implemented in Python or Scala only.

What we’re looking for in your code are readability and easy maintenance. We want to see a code that reveals its intent to the reader and follows best practices. To accomplish this you can use any paradigm you want.

You also have to make sure the code really works and use any tool or technique you need to accomplish this.

The Problem:

1. 1. You should start with a two-dimensional grid of **50x50** square *cells* and each of these cells is either *alive* or *dead*. You can start with a grid showing dead cells (we use 0 for example) and living cells using (we use 1). The initial configuration of living cells in this grid is arbitrary and should be random for every page refresh. Here’s a smaller 5x5 example:  
     
   00000  
   00000  
   01110  
   00000  
   00000
2. This grid is subject to changes on what is called a *tick*. When a grid “ticks”, these are the rules to determine the next state of the grid:
   1. Any live cell with fewer than two live neighbors dies (underpopulation).
   2. Any live cell with two or three live neighbors lives on to the next generation.
   3. Any live cell with more than three live neighbors dies (overcrowding).
   4. Any dead cell with exactly three live neighbors becomes a live cell (reproduction).

Try to look at the first state of this grid and apply the rules above. The result will be the second grid, as shown below:

00000 00000  
00000 00100  
01110 - tick -> 00100  
00000 00100  
00000 00000

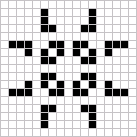
**That’s all!** Just to help you, here are images of some grids ticking every ~0.4s (white is dead and black alive). You can use them to check if your code is correct:











What we expect to receive:

1. Link to Github with fully functional development (not modified and not obfuscated) code.
2. If needed, please include a brief description of how to run the game.
3. Use Scala / Python
4. Write several unit-tests (not snapshots)

Good luck!