

HW1, Due Thursday, February 17, by 5pm.

You may submit your homework in class on Wednesday. If you submit it after class on Wednesday, it should be submitted to Prof. Dickerson in his office or to the slot on his office door: MBH 636.

This homework has both a written component and a programming component. You should begin the written component immediately. Before attempting the programming component, work through all three tutorials at the NetLogo documents website: ccl.northwestern.edu/netlogo/docs/

Written Component: Read about John Conway's *Game of Life*. (The course Syllabus/Schedule has links to three websites about the game. You can find more easily if you'd like.) Write a 2-page paper (Double-spaced, 1" margin, 12pt Times New Roman font) on **Life**. Address both the simplicity and complexity of the game and its rules, the interest it has generated, etc. Argue *yes* or *no* to the following questions and explain your answers. *Should Life should be considered a multi-agent simulation? Should Life should be considered an individual-based model?*

What to submit: A printout of your paper.

NetLogo Programming Component: Before writing your own models from scratch, this lab will give you an opportunity to experiment with, and then modify, NetLogo code for a very famous model: **Life**.

Prelab 1: Do all Tutorials 1 and 2 before class on Friday. Due Tutorial 3 before class on Monday (and before moving to Prelab 2).

Prelab 2: Go to the models library. Under Computer Science/Cellular Automata open the model for **Life**. Experiment with the model. Try random starting configurations. Make your own starting configuration. Try constructing a “glider”

Lab: Open the procedures tab of the **Life** model. You will see the following code in the `go` procedure.

```
ask patches
  [ ifelse live-neighbors = 3
    [ cell-birth ]
    [ if live-neighbors != 2
      [ cell-death ] ] ]
```

It will take a few weeks of class to go over all the aspects of this code, but here is a quick summary of the logic:

- If the number of live neighbors of a given cell is 3, the cell becomes alive whether it was alive before or not.
- If the number of live neighbors is not 3 or 2, the cell is dead whether or not it was alive before.

- If the number of live neighbors is 2, a cell remains in the same state: if dead it remains dead, and if alive it remains alive. In other words, no action is taken.

This code could be rewritten in the following way that would have the equivalent result. This way is more wordy, but might make the logic easier to understand and easier to change.

```
ask patches
[ if live-neighbors = 3
  [ cell-birth ]
  if live-neighbors = 0 or live-neighbors = 1
    or live-neighbors = 4 or live-neighbors = 5
    or live-neighbors = 6 or live-neighbors = 7 or
    live-neighbors = 8
  [ cell-death ]
]
```

Replace the original code with this new code, exactly as written above. Make sure it still works. Then change the rules of life (when a cell dies and when it gives birth) and experiment with various new sets of rules.

What to submit:

1. A brief description of one of your rules that you experimented with. (When does a cell give birth? When does it die? When does it stay the same?)
2. Your new code (not the entire code, just a past of the text from the important section of code that you changed).
3. A screen snapshot of a starting configuration and the configuration after a certain number of ticks have passed of what happens with the rule given in parts 1 and 2 above. (Specify how many ticks passed.)