A1 Problem Statement

Overview & Objectives

The Game. In this assignment, you'll build a browser-based version of John Conway's Game of Life, a cellular automaton that simulates the evolution of an organism using a simple deterministic rule.

Writing code in JavaScript. The main purpose of the assignment is to give you practice writing high-quality code in JavaScript. This includes not only the superficial aspects of coding (such as neat formatting, judicious commenting, having functions of reasonable size, and so on) but also the more subtle aspects that have a major impact on how easy the code is to understand and modify, and which tend to require more sophisticated techniques.

Idioms. In particular, you should use the idioms you've learned in class, notably abstracting iteration with functionals and using closures to control access to state.

Localizing design decisions and avoiding repetition. A good check that your code is well structured is to consider whether each of the key design decisions (such as the shape of the board, or the rules for playing the game) is expressed at a single point in the code. Also, you should avoid repetition and redundancy so that changes even to small features (such as the dimensions of the board) can be made in just one place.

Design Reflection. Although this assignment is focused on coding, the theme of the class as a whole is design. Each assignment will therefore involve conducting some design reflection, and this assignment will give you your first experience doing this.

Specification

Your pair's (or individually) task is to build a browser-based version of John Conway's Game of Life. Users should be able to select one of a collection of preset starting states, or create an arbitrary starting configuration, and start and stop an instance of the animation while it is running.

Provided code. To spare you the effort of implementing a user interface (which we will cover later in the class), we are providing you with some starter code that sets up the visual board and controls for you. Your task is to write the code that maintains the internal representation of the game, applies the rules, and sends updates to the visual board. Implementing this code involves providing the bodies of some functions in a given file. Of course, you should not constrain your code to be entirely within those functions; they will likely make calls to other functions that you will declare. **To open & manually test your implementation in the browser** in Chrome go to File > Open File… > Select & open index.html from finder and this will show you the game of life UI.

Client side code only. Your implementation will be loaded as a web page, and run without a server. This project requires client-side code only. You should only be writing JavaScript that runs in a web browser, and there is no need to provide any server-side code to generate these pages.

Commenting and testing. Your code should include succinct specifications of functions and classes, should be commented appropriately. Testing graphical behavior is hard, and we don't expect you to do this automatically — eyeballing the output is enough.

Design Reflection. (max 1000 words) In your design reflection, you & your partner should describe:

(a) The key design decisions you made, and how you achieved localization and avoided repetition.
(b) How you exploited functionals in your code, and why they improved the design.
(c) Some alternatives you considered to your design and why they were not chosen.
(d) Some limitations of your design and how they might be addressed.
(e) Some comments on the ethical implications of the project (see hints below).

Deliverables

- The entire code of your project, comprising the starter code with your code inserted in the file *internal.js* (which provides several skeletal functions that you should complete).
- A file called r*eflection.md,* in the top-level directory of the repo, that contains the design, ethical reflections. Give each section a header and in the header, put the partner responsible for that section in parenthesis
- If you worked with a partner, each partner must fill out the prompts of the Assignment repo's *critiques.md* file.
- If you worked individually, fill out the sections for Partner 1 in *critiques.md* except the Partner Critique & Authorship sections.

Collaboration Policy for Partner Assignments

Please review and follow the collaboration policy for partner assignments on the course syllabus.

Remember that both partner's should engage with all parts of the assignment (code implementation, design, and reflections) and divide the work equally.

Grading

See the accompanying rubric.

Hints

Design Reflection. Keep your discussion succinct, clear and to the point. Including irrelevant comments or repeating yourself will lead to a lower grade. Some design decisions to consider (also called "dimensions" or "axes" of a design) include: what data structures you used to represent the game board; how you chose to iterate over cells; how the board was updated or replaced at each step; how presets were represented and loaded.

Ethical Implications. The last part of the design reflection asks you to comment on some of the ethical implications of your project. You are free to consider any moral or ethical issue that you think is relevant, so long as what you write is coherent, thoughtful and to the point. Recognizing that the Game of Life has been taken by many computer scientists as a metaphor for biological and evolutionary processes, or even for population control, you might consider how this metaphor influences, positively or negatively the way we view the real world and our role in it. Or you might examine to what extent you just implemented the rules without considering their meaning, and whether this is typical of how programmers operate in corporate settings.

Documentation of Public Interfaces: You are not required to use any particular JavaScript annotation tool (if you'd like to, JSDoc is a popular tool), nor any particular styling of comments. Recall from 6.031 that to document your interfaces, you wrote brief, stylized comments for each function and, for each module, a very brief summary of what the module provides.

Use of Functionals: This assignment provides many opportunities to use functionals, as taught in lecture, to eliminate duplication in the code and to make it more succinct and more elegant. In particular, you may want to think about how you iterate over the neighbors of a cell, and whether this can be abstracted. One of the former 6.170 TAs (Harihar Subramanyam) made a guide about common misuses of functionals. It's highly recommended that you read it before starting the pset.

Additional References: The Wikipedia article on Game of Life; an implementation using an external graphics library, with simple source code (but you can do better!); a video of Conway talking about the game; a longer discussion of the game, its properties, as well as different objects and their properties (including stationary, oscillating, and gliding objects).

Edge Behavior: When live cells reach the edge of the grid, there are a number of ways they could behave, including: cells off the board could simply all be "dead" cells, or "live" cells could wrap around the edge of the board. Any behavior is acceptable, so long as your implementation is consistent and reasonable.

Markdown: When writing your *reflection.md* file, check out this cheat sheet for information on how to style markdown.

Testing: Although you will not be assessed on writing tests (graders will not run them), if you choose to write tests, we suggest using Jest.