3. WRITTEN RESPONSES

3 a.

3.a.i.

This program presents a interactive cell simulation following the rules of John Conway's Game of Life, to serve as entertainment for a user.

3.a.ii.

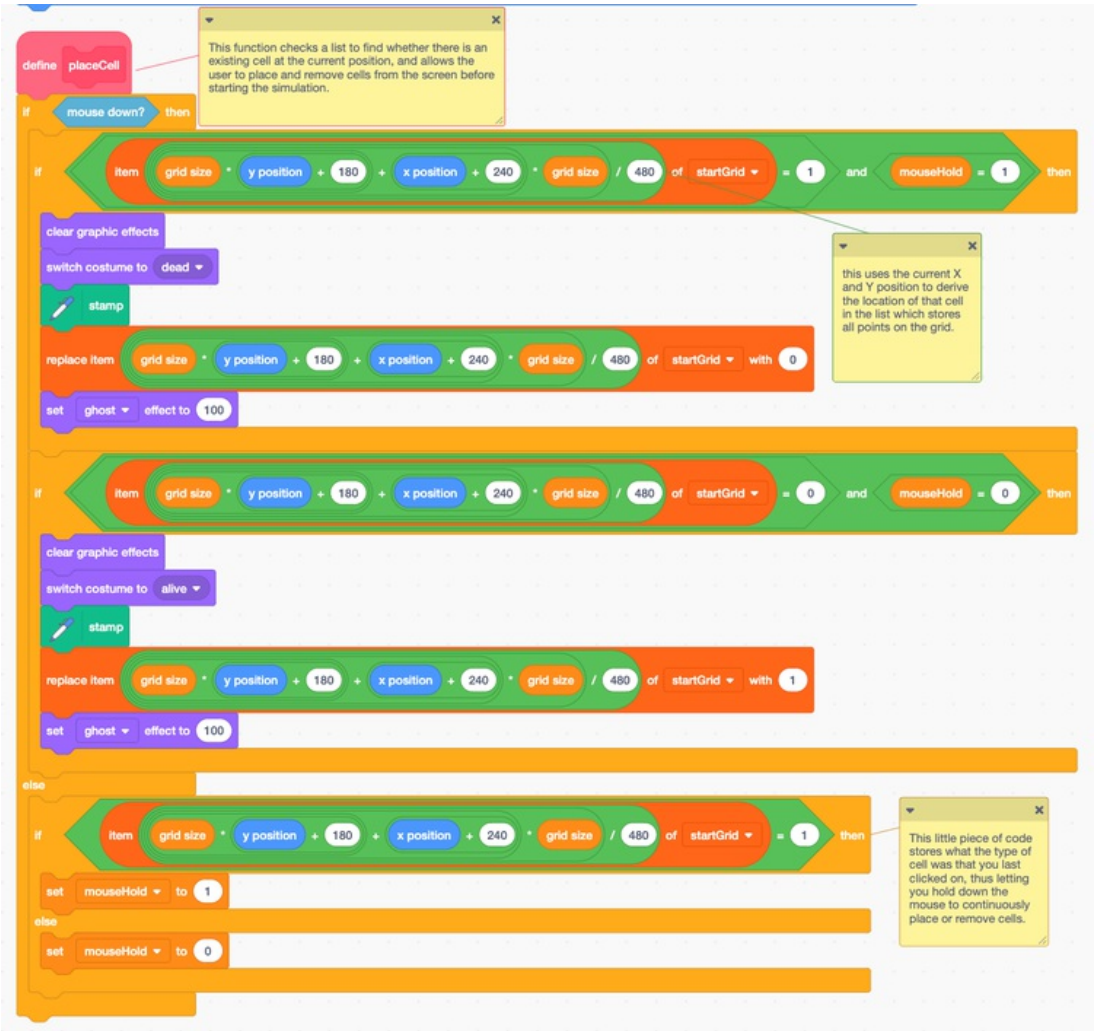
The program allows a user to place and remove tiles on a grid, which are then progressed programatticaly by the rules of Conway's Game of Life, with the program updating the screen after every generation. The user is also at all times allowed to pause, restart, and reset the program, editing and replaying the simulation.

3.a.iii.

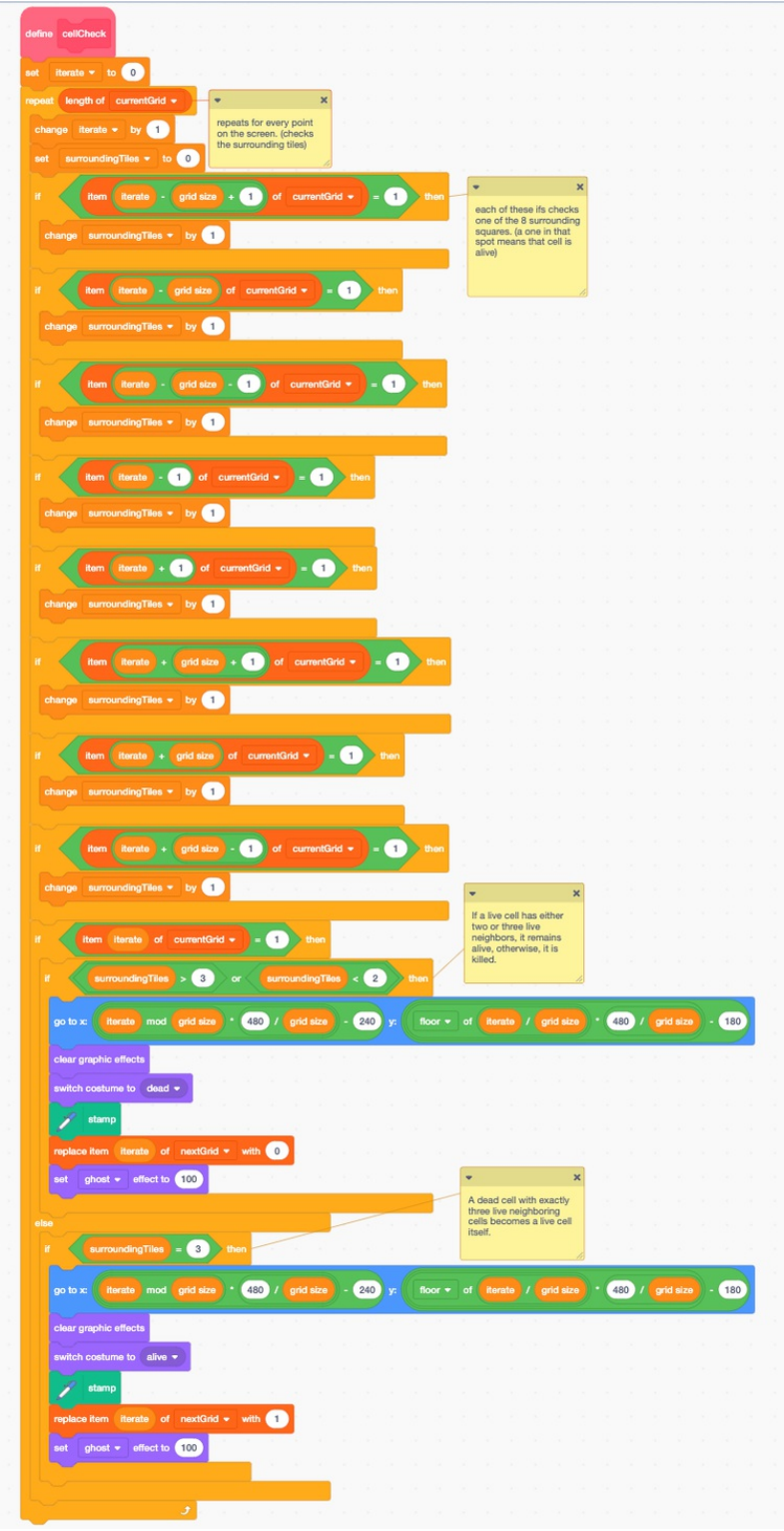
The inputs of the program are the keys space, p, and r, as well as a user's mouse clicks/position. The user's mouse position and clicking are taken in, with an output as a placed or removed cell on a grid, followed by the progression of those cells through simulated generations, paused, reset, and cleared by the keys p, space, and r respectively.

3 b.

3.b.i.



3.b.ii.



3.b.iii.

There are three lists used her, startGrid, currentGrid, and nextGrid. There only exist multiple so as to be recalled even after edits have been made. In this particular instance, the start and current Grids are being used, though the two are essentially the same list as currentGrid is at this point copied exactly from startGrid.

3.b.iv.

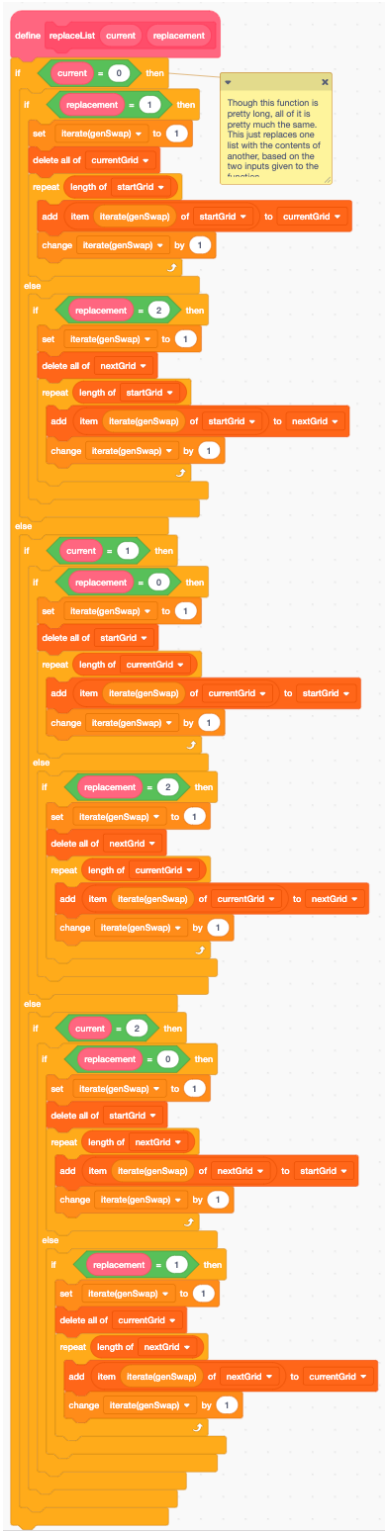
Each item in these lists stores the status of one of the 10,800(120 x 90) possible cells on screen. A 0 being dead, and a 1 being alive. From the position in the list can be derived the coordinates of a cell, and vice versa.

3.b.v.

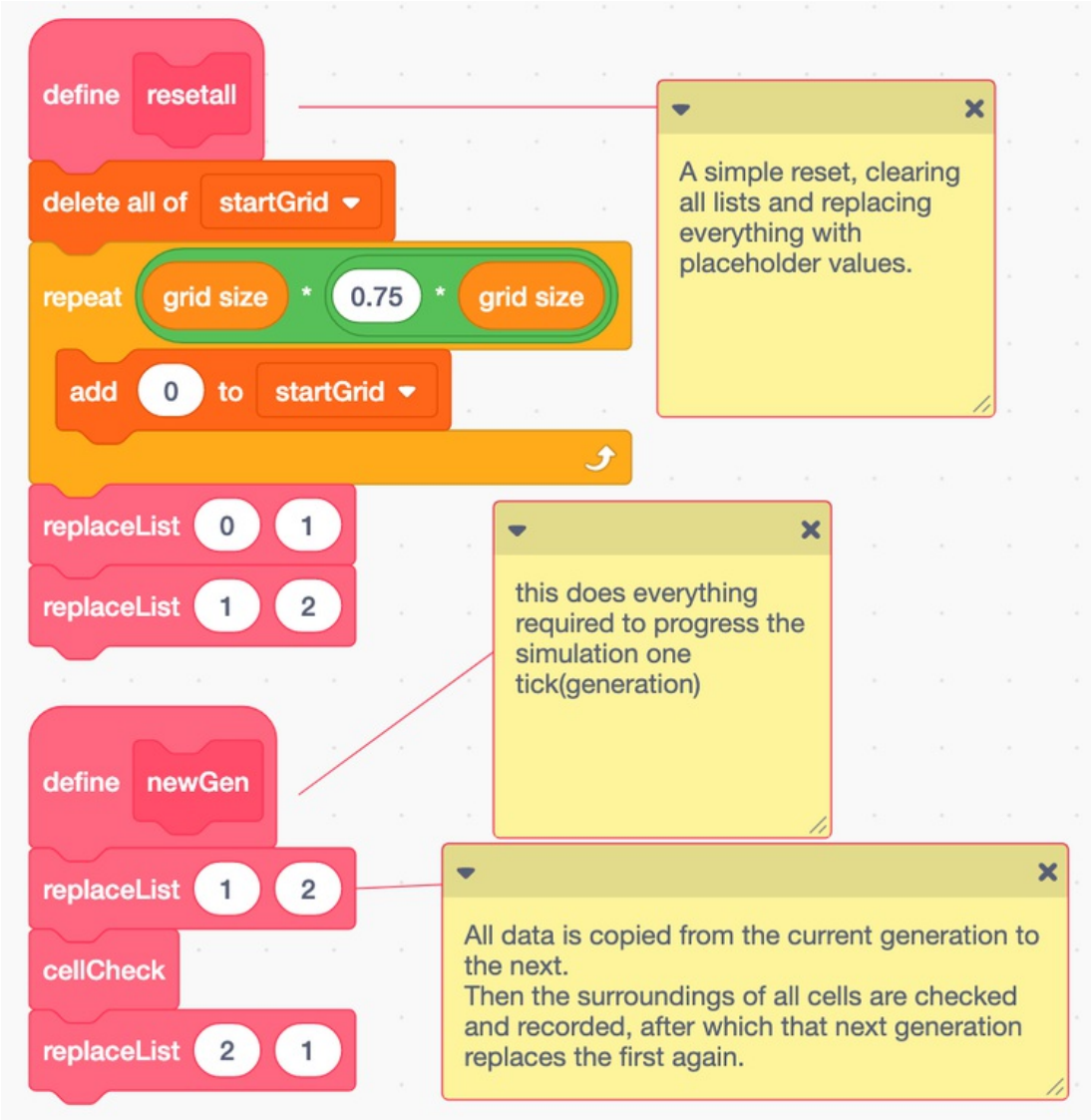
10,800 items must be kept track of, with a single position corresponding to a single cell, and a way to find one from the other. To do so with variables or some other method would be a monumental task, and very inefficient, so the simplest and easiest way is to do so is with a list. A list also consolidates all that information in one place, making it easier to use and keep track of. In a list, it is also easier to replace one individual item than it would be to replace a single character in a variable or other tracking method.

3 c.

3.c.i.



3.c.ii.



3.c.iii.

This procedure takes as inputs identifying numbers for two lists. The first will be copied over to replace the second. It is used in the program to store all cells in a generation, replace a different list with those same cells, and be able to recall a previous set of points through the copying of lists, so this allows for easy duplication and saving of all grid positions.

3.c.iv.

The algorithm uses nested if-else statements to identify which list will be copied from, by looking at the first parameter. Inside the if statement matching that parameter, are another set of if statements to determine where the list should be copied over to. This could be adapted to any number of lists, but is in this case just three, meaning two if statements within each of three nested if-else, as of course a list can not be copied to itself. Inside the innermost if statements is contained a loop with an iterating variable to copy over the items from the first list to the second one by one. The loop repeats for the length of the first list, but before the loop, the second list must be cleared in case they don't have the same number of items originally.

3 d.

3.d.i.

First call:

The first call is in a function used to reset everything, used right at the beginning of the program or whenever the user chooses. This call takes inputs of 0, 1 being the current and replacement parameters respectively.

Second call:
The function is called again during the main generation function, with parameter inputs of 2, 1 , being the current and replacement parameters respectively.

3 d.ii.

Condition(s) tested by first call:
This call checks which list is being copied from, and where to. The function determines which of 0, 1, or 2 each of the two parameters are. With parameter 'current' being 0, the first if will be entered, where, with parameter 'replacement' as 1, the first if will again be entered.

Condition(s) tested by second call:
This call checks which list is being copied from, and where to. The function determines whether the two parameters are each 0, 1, or 2. As parameter 'current' is 2, and 'replacement' is 1, the first if will fail, as will the second, and the third entered. Within that, the first if fails again, and the second is entered.

3.d.iii.

Results of the first call:
The entirety of the list "startGrid" will be copied over to replace list "currentGrid"

Results of the second call:
The entirety of the list "nextGrid" will be copied over to replace list "currentGrid"