**Robert Navarro Design and Reflection**

**Preprogramming thoughts and design:**

* Initialize an array of blank spaces to begin.
  + This array will have 40 columns and 20 rows.
  + Give the user the option of:
    - Fixed oscillator
    - Glider
    - Glider cannon
* Once an option is selected allow the user to set a desired location. (Not quite sure how the user will select this but I will play with different ideas).
* Display starting position for the option selected.
  + Blank cells will be filled with and “x” to represent cells that are alive
* With a starting position selected the program will need to loop through the array checking each cell.
  + Occupied cells with 0 or 1 neighbors will become dead.
  + Occupied cells with more than 3 neighbors will become dead.
  + Occupied cells with 2 or 3 neighbors remain alive.
  + If a dead cell has exactly 3 occupied neighbors it becomes alive.
  + These changes need to be calculated for each generation
* Calculating changes
  + Creating a second array seems like the easiest way to display changes
  + The first array will be scanned and the second array will be occupied with the updated cells.
  + There will need to be some sort of switch in place that allows for the display to switch from generation to generation.
* Scanning the active array
  + Scan through the active array and check surrounding cells for “x” or “-“ values.
  + Use the rules for the game of life and determine if the cell for the next generation is alive or dead.
  + Update the work array with the value for the next generation.
  + Copy the values from the work array to the active array.
* Displaying the values
  + Have some method for asking the user what shape they would like and where to display it.
  + Use two loop to display the array values.
  + Have another loop that determines how many generations will run.
  + Have another loop that checks if additional runs are needed or if the program should exit.
* These are my initial thoughts and I am sure they will change as I create my program.

**Post Coding Discussion**

I have just completed building and testing my program and I think I was able to stick to my original design ideas fairly well. One of the thoughts that I had after first making this design was to possibly try and build classes for this program. However, I couldn’t come up with a simple way to do this so I just stuck with one file.

I created several different functions to try and keep things organized. I came up with a copy function that allowed me to easily copy one array to another. This was especially helpful when I was checking if each cell was alive or dead. I would scan the active array and then place the appropriate value into the work array. After I had completely worked through the active array I copied the work array back to the active array.

I did have some issues at first trying to figure out how to copy each array. At first I was doing it several times in each function so I decided to create a function in order to keep things more organized. It also took me a few attempts to make sure that each cell was being correctly checked. I tested this in a small array at first since it is easier to check on a smaller scale. Once I confirmed that it was working properly I started working on the larger array.

When it came to displaying values on the larger array I ran into a few issues. I decided to go with the ghost cells idea that someone mentioned on canvas. This involved have cells that would not be displayed when I was outputting the array to the screen. The first time I read through the instructions I missed the part that said that the glider should disappear off the screen. This caused issues with the glider becoming a square at the bottom of my display screen. I ended up adding a few additional rows and columns and this allowed the glider to disappear off the screen.

Coming up with the method to display the different shapes on the screen took a while. I decided on having prompts come up on the screen to help the players choose the options that they wanted. I then called function for the desired display value and set the active array with the first generation value. I also made sure to include some input so that the user could place the shape within the 20x40 bound on the screen.

Having the display run correctly took a few times due to making sure that the first generation was correct. I quickly discovered that if the first generation was not in the correct location, the whole sequence would not work correctly.

**Testing**

I decided for testing it would be important to test all of the edges. One issue I ran into during testing was that the glider was not disappearing off the edge. This was due to not treating the 20 x 40 grid as if it was on an infinite plane. I was originally handling the area as if it was only 20 x 40, which caused the glider to turn into a square at the bottom of the grid. Once I realized my error the glider disappeared as expected.

For further testing I placed the shapes in different areas of the array. I tested all 3 design choice at random areas on the array and found things to be working as expected. I also made sure to test values that were valid and found that the program ended correctly if an incorrect value was entered.