Requirements:

- Cells that are alive stay alive only with 2 or 3 neighbors
- Cells that are dead become alive with exactly 3 neighbors
- Dying cells cannot be saved
- Need a 80 X 22 2D array
- Edges cannot be treated as only having 5 neighbors
  - They must be treated like they have 8 neighbors with 3 "invisible"
- User needs several patterns – oscillator, glider, and glider gun

Design:

Initial thoughts after reading assignment and number grids for hard coding in the patterns:

See initial thoughts pdf

Pseudocode based on initial thoughts

```
Create two 2D arrays both initialized with 0
Do
    Ask user what pattern they want to add
    Ask user what position they want to add it to
    If position will work by making sure it won't over lap with other patterns
        Add the pattern to the first array by call appropriate add function
While the user wants to add more patterns
Do
    Set up a for loop which will iterate for 100 generations
        Current array is the for loops counter % 2, where 0 is the first and 1 is the second
        Set up a for loop to loop through the rows
            Set up a for loop to loop through the columns
                Print the cell
                If the current cell has a value greater than 9 that does not equal 12 or 13, subtract 10 and call the adjust function with -1
                Else if the current cell has a value of 3 add 10 and call the adjust function with 1
                Add the cells value to the same cell on the future array
```

```
                    Set the cells value to 0

        Ask if the user wants to continue
    While the user wants to continue

Adjust function: takes a 2D array – future array, and 3 ints – row, column, and add value
    Create 2 ints – one for row number, the other for column number
    So set up a for loop for rows initialized at row – 1 until counter is greater than row + 1
        Set up a for loop of columns initialized at column – 1 until counter is greater than column + 1
            If row counter doesn't equal row or column counter doesn't equal column, then add add value to that cell

Add oscillator function – takes a 2D array and 2 ints – row and column for the top left most cell
    Hard code in the cells numbers based on the grid in the initial design
Add glider function – takes a 2D array and 2 ints – row and column for the top left most cell
    Hard code in the cells numbers based on the grid in the initial design
Add glider gun function – takes a 2D array and 2 ints – row and column for the top left most cell
    Hard code in the cells numbers based on the grid in the initial design
```

Testing

I plan to test the program by first not using a graphical display cells, but rather a numerical display. Since my cells are represented by numbers I can see what is happening to each cell at each generation. This means that I will be able to see that neighboring cells have the right numbers and that if a cells has a value of three that 10 is added to it, or if the cell has 10, 11, 14, 15, 16, 17, 18 that 10 is subtracted from it. And when either of those things happens the values of the cells all around it are adjusted properly. In the case of the edges, I plan to add edge lines that will not be printed to the screen. I plan to test this with the glider pattern and make sure that the glider moves off the screen and doesn't cause problems.

Reflections

I had never heard of the game of life, so my first challenge was figuring out what it was supposed to look like. After I went to Wikipedia, I read a line about vectors and order pairs (not sure what they intended to say, but…), this caused me to think about the cells as a number. Once I had that I started my initial thoughts document and let that expand. When I got done with my edge case thought, I decided to test my idea of the gird. This led to my next problem, how do I add/subtract 10's and 1's and not affect the current numbers so that they will still do what they need. This led me to using 2 arrays, which was the last part of my initial thoughts – outside of getting my number grids for the patterns.

At that point I was pretty confident I knew how to do the problem (partly because I have done grids with invisible lines). However, when got to coding the 2D array I found it clunky for me. I had read about pseudo multidimensional arrays before so I decided to use that instead. The next change from my design was that instead of hard coding every number besides 0 from my number grids, I decided to simply add 10 for my live cells and call adjust to add 1 to all surrounding cells. I then decided to create an add cell function so that I only had to type one line instead of two.

After that I ended up adding parts so that I didn't have to click through or watch and empty/static screen. If the visible board clears of all live cells, or if all the cells become static, then the program states this and quits.

Now that I am done, when I think back on it, I think my biggest problem was actually implementing a menu. It is still not exactly what I would like, but spending over 12 hours on it is my limit, and it works. I personally would have liked for dead cells to be neighbors of multiple patterns, but my clear board function and availability testing would not allow for that.