James Sandoval - CS 162

Assignment 1 – Conways' Game of Life Design, Testing, and Reflection

Design (Psuedocode):

```
Define the Conway class
    Create two Private member matrices:
        - World matrix
        - Mirror matrix
        - Define two variables for x and y coordinates
    Public:
        Constructor:
        -   Define the two matrices size
        -   Initialize the two matrices

        Setter Functions for x and y coordinates

        Function to print the matrix
            Print the matrix – this includes formatting
            (This will be slightly smaller than the actual array – to simulate
            the appearance of a window on an infinite world.)
        Function to scan through the array
            Iterate through the entire array, implementing rules
            as defined in Conway's game of life
                Determine Whether or not a cell will live or die
                    Copy New cell value into mirror matrix
            After old array has been scanned in its entirety, new array is
            copied into old array (replacing its contents).

        Function for Oscillator
            Pass x and Y coordinates as parameters
            Set living cell values into position for pattern
        Function for Glider
            Pass x and Y coordinates as parameters
            Set living cell values into position for pattern

        Function for Glider Cannon
            Pass x and Y coordinates as parameters
            Set living cell values into position for pattern
```

```
Main Function

Create a Conway class object

    Greet the user
    Ask the user to choose a set of x and y coordinates
        Input x
        Input y
    Print a menu asking the user to choose what pattern they would like to use.
        Call the specific pattern function

    Use a loop to process generations
        Call the scanWorld Function
        Call the printWorld Function
        Use the Usleep function to delay until next print process
        Clear the current screen output
```

<u>Testing:</u>

Due to the fact that the game is a zero-player game, the user has limited inputs and functionality should be easy to plan and execute given any user defined variable.

Input:	**Result :**	**Expected Result:**
Menu Selection: 1-3	Program runs, and associated Pattern function is called.	Program runs, and associated Pattern function is called.
Menu Selection: < 1 or > 3	Program quits, error is printed.	Program quits, error is printed.
User inputs x coordinates (1-30)	program accepts coordinates and pattern is implemented.	program accepts coordinates and pattern is implemented.
User inputs y coordinates (1-20)	Program accepts coordinates and pattern is implemented.	program accepts coordinates and pattern is implemented.
If user inputs a character at any prompt.	Program quits, prints invalid input.	Program quits, prints invalid input.

Reflection:

Writing this program was definitely a great learning process and gave me the opportunity to explore C++ and coding in a way that I haven't done before. It was an awesome feeling seeing this program come into fruition despite the initial doubts of my abilities before attempting such a task. I know there's a ton of ways I could have implemented this program, other than the way I did, but this gave me the opportunity to create a plan, and execute, and I think I did mostly that.

When scanning through array, avoid counting edges or a segment fault will occur due to the fact that the outer edges of the matrix will be parallel to addresses that are not allocated. In order to mitigate this problem, the scanning function scans only the 2nd to outermost border of the matrix. I chose to use vectors instead of arrays because of the simplicity that vectors offered. I was initially able to use dynamic arrays, but quickly ran into a large number of memory access issues. After careful consideration and a great deal of trial and error I found that using vectors was simply a better approach for this project. In the future, I intend on developing a better understand between the relationship of pointers and arrays and how to properly implement them.

Due to the fact that this project is a zero player game, and the user may only select the starting coordinates and the type of pattern to implement there are very few variables that may function adversely. During the testing phase, when inputting coordinates, the matrix functioned according to design, however, if the user selected coordinates outside the viewing pane, the program would execute and the user would observe nothing. To mitigate this issue, I chose to implement a default x and y value if the user chose a range outside of the viewing pane. If this happens, the user is alerted, the value is printed to the screen, and the program continues to run.

In short, this was a great learning process, and I think I will greatly benefit from creating more designs and implementing them. I have a feeling I'll have a few more opportunities to practice this powerful cornerstone technique.