Assignment 4: Conway's Game of Life

Description

The *Game of Life* consists of a two-dimension grid of *cells*. Each cell is either *alive* or *dead*. Each cell has eight *neighbours*, which are the cells that are horizontally, vertically, or diagonally adjacent.

X-1, Y-1	X-1, Y	X-1, Y+1
X, Y-1	X,Y	X, Y+1
X+1, Y-1	X+1, Y	X+1, Y+1

At each step in time (or *tick*), the following transitions occur:

1. Any **live** cell with **fewer than two live** neighbours **dies**, as if by underpopulation.
2. Any **live** cell with **two or three live** neighbours **lives** on to the next generation.
3. Any **live** cell with **more than three live** neighbours **dies**, as if by overpopulation.
4. Any **dead** cell with **exactly three live** neighbours becomes a **live** cell, as if by reproduction.

The initial pattern constitutes the *seed* of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed; births and deaths occur simultaneously.

Reference

https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

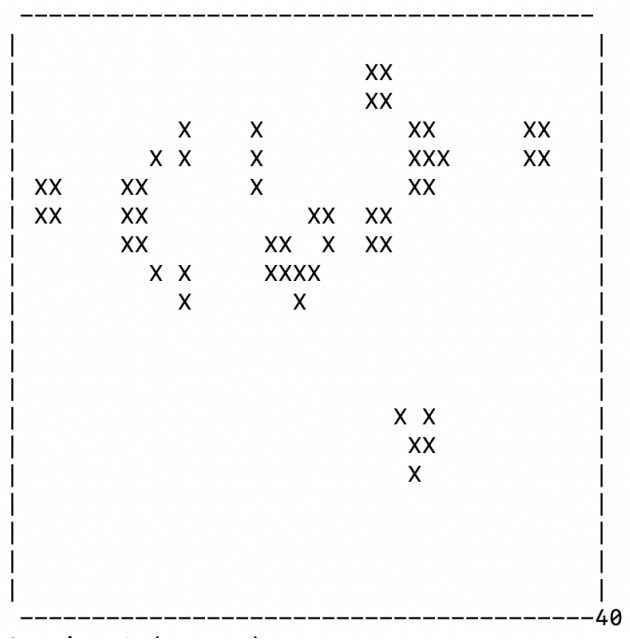
Now you have the description of the game but how do you program it?

Step 1. Write code to read in the seed (initial) grid.

- Set the size of your grid as #define statements (or you can make them dynamic – see the end of the assignment for more information).
- Open a file whose name is given as argv[1].
- Each line in the file will be a row in the grid and each number on the line represents a cell (or column) in the grid, e.g. if your grid was 4 X 4 then the following could be a seed:
  0 0 0 0
  0 1 1 0
  0 1 1 0
  0 0 0 0
- Read these values into a 2-D array. The grid can be an integer array or a character array and the values can be 1's and 0's (input format) or 'X's and spaces (display format).

Step 2. Write code to display the grid.

- Using the 2-D array populated by the previous code. Designate a live cell by an X and a dead cell by a space.



- Notice that there is a frame around the grid and in the lower right corner will be a number representing the tick (number of times that the grid has been evaluated).
- To see if your code is working for both reading and writing the grid, a number of seed files will be supplied with the assignment. These files will have the extension .seed.

Step 3. Evaluate all of the cells in the grid to see if there are alive or dead in the next tick.

- Since all of the cell calculations are happening "simultaneously" you need to have two arrays. One 2-D array represents the "world" (grid) as it is now and one represents the world after the calculations have been made.

Tick 0 (Seed) – Current Grid (for 1st tick)

		X		
	X	X	X	

Tick 1 – Future Grid (for 1st tick) and Current Grid (for 2nd tick)

	X	X	X	
	X	X	X	
		X		

Tick 2 – Future Grid (for 2nd tick) and Current Grid (for 3rd tick)

		X		
	X		X	
	X	X	X	

- In other words, evaluate a cell in the "Current" grid/array and put its new value (after applying Rules 1, 2, 3, or 4) in a "Future" grid/array. After displaying the "Future" grid/array, it becomes the "Current" grid/array.

Other Considerations

1. How can you simulate the grid changing over time but remaining in the same space on the screen?
   - Before you display the grid, you need to "clear" the screen. This clears all text off the screen and returns the cursor to the top of the screen. The C code to do this is:
     ```
     system("clear");
     ```

2. How can you stop each successive drawing of the grid from coming too quickly to the screen?
   - After each display of the grid, you need to have the system "sleep" before it displays the next generation of grid. The C code to do this is:
     ```
     system ( "sleep 0.25" );
     ```

3. After loading the seed, you should let the user see it before the process starts so as the user the following question:

   Start? (y or n):

   If the user answers n then terminate the program.

4. How many ticks should the program display?
   - Well it is hard to know what the seed will do so we should establish a number of ticks that will be displayed and then as the user if they want to continue with another set of ticks. The set of ticks will be the 2nd parameter on the command line. If this is missing then the number is set to 50. The program will go for either `atoi(argv[2])` ticks or 50 ticks before asking: Continue? (y or n): and then doing either another set of ticks or terminating the program.

5. Where is the seed file entered?
   - The name of the seed file is the 1st command line parameter (`argv[1]`).

6. What do I do when I get to the edge of the world?
   - At the edges of the grid, do not consider cells that are "not" in the world (ignore them when calculating the new world).

7. What size (dimensions) should I make the world?
   - Please make your world 20 rows by 40 columns.
   - But what if I want to make this dynamic (set as a command line parameter). You will get 2 bonus marks (2 full marks on your final grade).

8. Do I need to use functions?
   - No but they would be nice ☺

9. What should I do if the grid/world does not change from one tick to the next?
   - If that happens (and it does) you should terminate the program.

Example Command Lines

```
$ ./cgol diehard.seed 20
```

- Seed file is named diehard.seed
- The time interval (number of ticks) is 20.

```
$ ./cgol R-pentomino.seed
```

- Seed file is named R-pentomino.seed
- The time interval (number of ticks) is 50.