Assignment 1 – Program Design and Testing

Goals-

Coding review

Convert requirements (i.e. the rules) to a software design

Think about testing process, before and after writing the program.

Implement Conway's *Game of Life*

Conway's *Game of Life* is a standard example of a cellular automaton. This means that you have an array or matrix of cells. Each turn or generation the value of each cell may change based upon its neighbors. The rules are:

Each cell has eight neighbor cells. The neighbors are the cells directly above, below, to the right, to the left, diagonally above to the right and left, and diagonally below to the right and left.

1. If an occupied cell has zero or one neighbor, it dies of loneliness.
2. If an occupied cell has more than three neighbors, it dies of overcrowding.
3. If an empty cell has exactly three occupied neighbor cells, there is a birth of a new cell to replace the empty cell.
4. Births and deaths are instantaneous and occur at the changes of generation.

A cell dying may help cause birth. A newborn cell cannot resurrect a cell that is dying. A cell's death will not prevent the death of another cell, say, by reducing the local population. NOTE- Think this through carefully before you start coding. If you get it wrong you can have an infinite loop as a cell dies, which changes a neighbor, which keeps it alive, which changes a neighbor, which …

You can also find much information about the game on the Internet. Much information…

You will design, implement, and test a program that runs the game of life. There are significant initial questions that you must address in your design. We will limit the visible "world" to 80 x 22 cells. That's a lot of characters to type in from the keyboard so don't make the user do it. There are numerous ways to address this problem that may involve hard coding arrays or using the keyboard. That's why it's called design! :-) But you need

to give the user some option to start with more than one arrangement in the grid. Remember that a starting configuration will **typically consist of less than a dozen cells.**

Probably the most difficult part of the design is what you will need to do to handle the cells on the edges. Remember this is a window on an infinite grid. You just can't stop the patterns at the edge as that may change the behavior. Specifically, any pattern should not change as the object moves out of the visible grid and the object should slide out of view. What this means is that your program must use the entire pattern or none of it as it exits the displayed area.

You will create your design document BEFORE you start coding. The TA will grade, in part, on how well your implementation matched your design. Remember, in your reflections document you can explain how you had to change the design because your first idea, just didn't work. Just explain what you learned. Simply stated, program design is identifying the problem to be solved, identify the inputs, specify the desired output, and then develop the algorithm(s) to convert the input into the output.

You must also describe how you tested your program. In this case you must demonstrate that the rules work for all cells. The edges will be a special case. You have 2 types of horizontal edges and two types of vertical edges. Each corner involves a different combination of these edges. Your reflections must described how you planned to test these cases and the results of the testing.

By the due date, you will submit your program, and the reflections document, which will discuss how well your design worked, and the results of the testing. You must also provide a makefile. All the files must be submitted in a zip archive.

Grading:

- programming style and documentation (10%)
- build the array/matrix providing a fixed simple oscillator (30%)
- the display allows the user to see the change(s) in the pattern (10%)
- provide a stop command (5%)
- allow the user to specify the starting location of the pattern (10%)
- add the glider pattern (5%)
- the glider should simply disappear off an edge and not disrupt other cells in the grid (10%)
- add a glider gun or cannon pattern (5%)
- reflections document to include the design description, test plan, test results, and comments about how you resolved problems during the assignment (15%)

HINTS-

1. Use the grading breakdown to plan your program. Use incremental development. Get one part working. Test it. Save a copy and continue working on the next step in your plan. This ensures you wil have something to submit!
2. On the Internet you will also find many descriptions and examples of stable patterns. Definitely take advantage of that for your testing!
3. There are only 4 rules that you apply to each cell in your array. There are emergent properties so testing logic errors may be difficult. :-)
4. For cells on the edges you need to handle the case that you are displaying a subset of an infinite grid. Get your program working first, ignoring the edge condition. Once you know your code is working then implement the edge algorithm(s). One option is to have some ghost columns and rows. You can continue calculating the movement, but just not display some of the cells.
5. As one website stated, the *Game of Life* is one of the most programmed games in the world. Be very careful about borrowing any code, or ideas you see in someone else's code. As always, any code submitted must be yours and yours alone.