The Game of Life

EE312

Due: Friday 2/15/19 at 6:00pm
100 points

For this assignment you will be implementing the classic cellular automotan Conway's Game of Life.

Rules

- A new cell is born on an empty square if is surrounded by exactly three neighboring cells
- A cell dies of overcrowding if it has four or more neighbors
- A cell dies of loneliness if it has zero or one neighbor
- Any live cell with two or three live neighbors lives, unchanged, to the next generation.

The initial state of the game will be held in a text file. The format will be sequences of ones and zeros (characters). A '1' means that a cell is alive in that square. You will determine the number of columns in the grid display by reading the first line of the text file and seeing how many characters are in the string. You will determine the number of rows in the grid by counting the lines of data. You will need to know the number of rows and columns in order to dynamically allocate memory for the subsequent iterations of the world.

Here is an example of what a world might look like. This grid may be of arbitrary size, but the number of columns will be the same for every row.

00000000100000001010
00000000010000001001
11100000010100000010
10100100101010101010
00101010010010101000

When printing out the status of the grid, print one character to standard output per cell in the automaton. Use an asterisk (*) to show that a cell is alive and a period (.) to show that the cell is dead.The driver program will clear the screen between generations.

The name of the input file and the number of iterations will be passed as parameters to the main function. Here is an example of how the program will start:

<linux_prompt>% ./life starting_grid.txt 250

Where the name of the executable program is "life", the input file is "starting_grid.txt", and the number of iterations is 250.

NOTES:

- You must do this program by yourself.
- This program must be done using a Linux environment. Note: Your code must compile and run on kamek.ece.utexas.edu.
- Please see the code in this example to clear the screen.
- The program must be modular, with significant work done by functions. Each function should perform a single, well-defined task. When possible, create re-usable functions. Do not write trivial functions such as a function to read a single value from an input file.

- You must place appropriate functions in a library. You will be given a starting life.h file. Don't change the function definitions in this file. You will also be given the life_driver.c source file. Only make the small modifications as mentioned in the comments.
- You must name your C source files life.c, life_driver.c and life.h (make sure you comment the header file well).

Turn in: life_driver.c, life.c, life.h (you can use the makefile that we are providing).

Upload: Turn in a zipped file named prog02_xxxxx.zip where xxxxxx is your UT EID to Canvas.

Be sure to follow the style standards for the course.

rlp 1/29/19