Assignment 4 pdf:

Description:

In this assignment, I will be recreating the game of life in C. The game of life is a simple game where there is a 2D grid of “alive” and “dead” cells, that theoretically span infinite distances. Though our boards will not span infinite distances due to the fact that the memory of a computer is finite. The rules of the game are simple if an alive cell has two or three alive cell neighbors it survives. If a dead cell is surrounded by exactly three alive neighbors it becomes alive. All other cells die either by overcrowding or loneliness.

Files to include:

- universe.c implements the Universe ADT
- universe.h specifies the interface to the Universe ADT. This file is provided and may not be modified.
- life.c contains main() and may contain any other functions necessary to complete your implementation of the Game of Life

Pseudo code:

universe.c:

- FUNC: Universe *uv_create(uint32_t rows, uint32_t cols, bool toroidal)
    - Creates a struct of type universe
    - Will use a pointer to an array of pointers to create a two-dimensional matrix
    - Then set the toroidal boolean value to the correct value;
- FUNC: void uv_delete(Universe *u)
    - Uses a for loop and free() to clean up to all the column arrays
    - Finally, use free() for the final row array
- FUNC: uint32_t uv_rows(Universe *u)
    - Simply return the value stored for the number of rows
- FUNC: uint32_t uv_cols(Universe *u)
    - Simply return the value stored for the number of columns
- FUNC: void uv_live_cell(Universe *u, uint32_t r, uint32_t c)
    - Set the value of arr[t][c] to true
- FUNC: void uv_dead_cell(Universe *u, uint32_t r, uint32_t c)
    - Set the value of arr[t][c] to false
- FUNC: bool uv_get_cell(Universe *u, uint32_t r, uint32_t c)
    - Return the value at arr[t][c]

- FUNC: bool uv_populate(Universe *u, FILE *infile)
    - Use infile to read an external file
    - Use the first line's row and column values to create the universe struct
    - Loop for the remaining pair values in the file until there is nothing
    - For every value pair set the cell at the coordinates to alive
    - If it is out of bounds return an error message
- FUNC: uint32_t uv_census(Universe *u, uint32_t r, uint32_t c)
    - For the coordinate value given start checking the value of arr[t - 1][c - 1] to arr[t + 1][c + 1]
    - If the toroidal is set to true then when on edges loop around to the other side of the board (theoretically I could increase the size of the array by 2 rows and columns and then store the other value on the other side but it seems it would complicate the program too much) use if statements to check if you are on the border
    - Then loop through all possible positions and check if the it is alive and increment the counter
    - Return the counter
- FUNC: void uv_print(Universe *u, FILE *outfile)
    - Use two nested for loops to iterate through all values of the matrix and print out each cell's value

life.c:

- Use a simple getopt program structure
- Use a switch and case statements
- Have outside vars to correctly set the appropriate values
- At the bottom have the logic that actually runs the program
- Have a for loop that runs as many times as the number of generations
- Have two nested for loops that iterate over all the values and then check their number of alive values
- If the cell is alive and has 2 or 3 alive members nothing happens if it was dead and has 3 alive neighbors it becomes alive record these incidents into an array
- Set all values to dead and then set all values from the array to alive
- At the end print out the result