WRITEUP ASGN4

Cruzid: acristea

February 2023 - CSE13S

1 Ncurses Library

The ncurses library allowed me to create a separate screen to print things, including the different generations of Game of Life. To begin you need its header file, ¡ncurses.h¿ You then need to initialize the screen and set the cursor to FALSE so it's hidden, only if case -s is not chosen. After each generation, you then want to refresh the screen, delay the screen by 50000 microseconds, then clear the screen, again only if case -s is not chosen. We need this because each generation our cells update, so the screen does too. We want to print the live cells for each row and column, talked about more in detail when explaining Game of Life, again only if -s is not chosen. Lastly, end the screen after every generation is done.

2 Compiling/Linking

Like other assignments, we had a header file universe.h that contained all our universe functions that we would implement in our test case file. We would make them in universe.c, using a struct, which is initialized in universe.c, and defined in universe.h. This made it very helpful to be able to easily link functions through files and organize my code. Since life.c was the executable file, it needed compiling, so in order to do that, we were tasked with making our own Makefile (just like our previous assignments). I utilized my Makefile to also compile additional test files that I use at the start of implementing my code. Furthermore, I used the file to reformat my code into clang-format which is one of the requirements for the assignment. One thing that I ran into was when I was running my tests on gitlab, there was a different requirement for clang which was a new update (so I had to update and format it according to the test).

3 Game of Life

The Game of Life is zero player game, meaning its evolution is determined by its initial state, requiring no further input. It can be played toroidal or non-

toroidal. Toroidal means the grid is torus-shaped, and technically nothing is out of bounds. If it checks for a neighbor and it's out of bounds we simply wrap to the other side. If it's non-toroidal we do nothing. We first have to make two Universes A and B. They begin with a set grid, determined by the first line in the input file. The live cells are inputted into the grid given by the next lines in the input file. In every cell of every grid, we should check to see if any rules are then being used. The rules are simple and are as follows:

- Any live cell with two or three live neighbors survives.
- Any dead cell with exactly three live neighbors becomes a live cell.
- All other cells die, either due to loneliness or overcrowding.

If the cells fall under any of the rules, we must change them accordingly. This is done using mvprintw from ncurses. At the same time as printing, we make every cell that lives in Universe B. It should then swap Universe A and Universe B and move on to the next generation. We repeat this process until every generation is done. Then we print the result into the terminal.

4 Conclusion

In conclusion, I learned how to use input files and output files in c. I further increased my knowledge of creating specific test case files. I learned how to use structs, which was essential to get the entire program to run. I learned the basics on how to use ncurses. I understand the importance of linking and compiling files and its significance when using multiple functions and variables. Lastly, I feel like this assignment allowed me to think outside of the box and use outside knowledge and logic-based understanding to get through the toroidal side of the functions.