Assignment 4
WRITEUP
Prof Veenstra
Nguyen Vu

In this assignment, I implement the Conway's Game of Life in C programming language and in this WRITEUP.pdf, I will talk about everything I learned while doing this assignment.

1. universe.c

1. I learned more about the type Struct variable. It is a "group" variable for me to put a lot of other variables inside and call those through the struct. For more details, last assignment also has the type Struct variable but it was given and we were also given a lot of instruction on how to work with it, pretty much everything we need to know about that specific struct. However, this time we were only given on what should be included in the struct and we needed to figure out how to work with it by ourselves.
2. in uv_create, I used calloc function to dynamically allocate memory for the first time because I used malloc function in assignment 3 and learned the difference between the 2. malloc has 1 parameter and doesn't initialize the allocated memory. calloc has 2 parameters and allocates the memory and also initializes every byte in the allocated memory to 0.
For more details, I did "Universe *x = (Universe *) malloc(sizeof(Universe));" to create a pointer to the struct called universe x and then used the given pseudo code to dynamically allocate the memory for the grid inside universe x.
3. I learned to loop and and forth of an array by using modulo in the next and previous functions.
For more details, the formula I used is

$$previous/next\_cell = ((max\_row/column)+(current\_position)+/-1)\%(max\_row/column)$$

This formula, in my interpretation, is we create a temporary array next to the current one, and add/subtract 1 based on which cell we want to get to, and if it gets to the temporary array that has the exact values as the current array, we remove the original array (with modulo) and use the new one, if not, then we remove the temporary array (with module) and keep the old one.

2. life.c

1. I learned how to get user file input/ouput with FILE data type, fscanf,

and fprintf functions. FILE data type is simply holding the file that the user wants to work with. fscanf scans data in file and we can customize what to scan. fprintf prints data into a file and we can also customize what to print. For more details, I made FILE *infile = stdin to make a pointer to the file type variable infile as user input, and FILE *outfile = stdout to make a pointer to the file type variable outfile as standard output. After that, I set infile/outfile to the files that the user want to use as input/output with getopt, fopen(infile, "r"), and fscanf(infile, "%d %d\n", &rows, &columns) to read the data in the input file and fopen(outfile, "w") with fprintf(outfile, ".") to print dead cells and fprintf(outfile, "o") to print alive cells to the output file. In the end, I used fclose(infile) and fclose(outfile) to close them.
2. I learned how to setup ncurses library to display my code on the screen for each loop with the help of the given pseudo code.
For more details, with the help of the pseudo code, I learned that initscr() is used to setup the screen, curs_set(FALSE) turns off the cursor, clear() to clean the screen, mvprintw() to print things on the screen, refresh() to refresh the screen, usleep(sleep time) to sleep for that amount of microseconds, and endwin() to close the screen. With the given pseudo code and helps from many TAs, I was able to understand where to put each components and how to use them properly to show my output after each generations of the program.