Assignment 4
DESIGN
Prof Veenstra
Nguyen Vu

1. universe.c

This first section will take about the design of the file universe.c

1.1. struct Universe

I simply initialize the type Struct variable called Universe with required variables to use in other functions.
Pseudo Code:
Make variable row and column
Make bool variable grid and toroidal

1.2. Universe *uv_create

I allocate the memory for the type Struct universe using the given pseudo code as well as setting the data inside the struct with the input file.
Pseudo Code:
Set row, column, and toroidal same as the input file
Given Pseudo Code:
uint32_t ** matrix = (uint32_t **) calloc (rows , sizeof (uint32_t *));
for (uint32_t r = 0; r ¡ rows ; r += 1) {
matrix [r] = (uint32_t *) calloc (cols , sizeof (uint32_t));
}

1.3. uv_delete

I remove all data from the grids and free the memory.
Pseudo Code:
Make a for loop to remove all data in columns and rows and the universe struct
Free memory of the grid and universe struct

1.4. uv_rows

I return the number of rows.
Pseudo Code:
Return the number of rows

1.5. uv_columns

I return the number of columns.
Pseudo Code:

Return the number of columns

1.6. uv_live_cell

I check whether the position of the cell is valid or not, if yes, make the cell alive.
Pseudo Code:
if (the given cell is valid) {
make cell alive }

1.7. uv_dead_cell

I check whether the position of the cell is valid or not, if yes, make the cell dead.
Pseudo Code:
if (the given cell is valid) {
make cell dead }

1.8. uv_get_cell

I check whether the position of the cell is valid or not, if yes, return the cell value.
Pseudo Code:
if (the given cell is valid) {
return cell value (bool) }

1.9. uv_populate

I loop through each line in the file and check whether the cell at the given position is valid, if yes then make it alive, if not return false.
Pseudo Code:
for (each row in the file that isn't end of file) { if (the given cell is valid) {
make cell alive }
else return false

1.10. uv_census

I check whether it is toroidal or not, then check the number of alive cells in 8 spots around the given cell following the rules for toroidal and non-toroidal.
Pseudo Code:
Initialize the counter variable
check if it's toroidal or not
check for alive cells around the given cell and +1 to counter if there's an alive cell
use previous and next function for toroidal (will be discussed later)

return counter

1.11. previous

Get the previous cell (loop around) in toroidal case using modulo.
Pseudo Code:
previous cell = ((max row/column) + (current spot in row/column) -1) % max row/column

return previous cell

1.12. next

Get the next cell (loop around) in toroidal case using modulo.
Pseudo Code:
next cell = ((max row/column) + (current spot in row/column) + 1) % max row/column
return next cell

1.13. uv_print

I print out the data to a file using fprintf().
Pseudo Code:
Use for loop to go through each rows
Use nested for loop to go through each columns
Print the cells according to it's status
Print new line at the end of each row

2. life.c

The first thing I do is initialize all necessary variables and do the getopt code (recycled from assignment 3) to get the user input. After that I check for input and output file and put errors if files couldn't be opened for the input was incorrect. Then I created 2 universes with the same size and implement the game. The screen output pseudo code were given. In the end I free up the memory of the universes, close the opened files and delete the universes.
Pseudo Code:
Set up all necessary variables
Get user input
Create 2 universes
Setup the screen
Loop through generations
Loop through each cell and check whether they are alive
If yes, print "o"

Check if there are enough cells alive around them with rules based on whether it's toroidal or not and set the cell to dead or alive accordingly
End the game after the last generation
Clear the memory and the data
Print the final generation to the output file

Give Pseudo Code for the screen:

Short ncurses example.

```c
#include <ncurses.h>
#include <unistd.h> // For usleep().

#define ROW 0
#define DELAY 50000

int main(void) {
    initscr();                      // Initialize the screen.
    curs_set(FALSE);                // Hide the cursor.
    for (int col = 0; col < 40; col += 1) {
        clear();                    // Clear the window.
        mvprintw(ROW, col, "o");    // Displays "o".
        refresh();                  // Refresh the window.
        usleep(DELAY);              // Sleep for 50000 microseconds.
    }
    endwin();                       // Close the screen.
    return 0;
}
```