Assignment 4 DESIGN.pdf
Yash Sharma

Description of Program:

This assignment requires us to implement abstract data types to create and implement the Game of Life in C. The abstract data type will provide the abstraction for a universe, a finite 2-D grid of cells. This program will work in finite memory and will be using structures, constructors, and functions to develop the Game of Life. In addition, this program will sort through for loops and return the rows and columns that contain live & dead cells. These cells will then be printed through a function and displayed on a custom rows & columns board set by the user interface. Once it runs, the program will generate the amount of live and dead cells on the Game of Life Board along with the forthcoming changes in the live/dead cells as the program undergoes multiple generations of changes.

Files to be included in directory "asgn4":

1. universe.c:
    - This file contains the implementations of the universe
2. universe.h:
    - This file contains the function interface to universe.c
3. life.c:
    - This file contains the implementation of all the functions required to build a life in the universe.
4. Makefile:
    - A file that formats program into clang-format and compiles it into program executables with make/make all from Makefile
    - Additionally, make clean from Makefile must remove compiler-generated files (such as the executables)
5. README.md:
    - This file describes how to use the program and MakeFile. It will also list and explain any command-line options the program accepts.
6. DESIGN.pdf:
    - This describes the design for the program thoroughly with pseudocode and descriptions.
7. WRITEUP.pdf:

- ○ This describes the program thoroughly with graphs displaying the integrated values for each number of partitions. The graphs will be generated using gnuplot and other UNIX tools. It will also include an analysis of the graphs and any lessons learned from floating-point numbers.

Pseudocode / Structure:

universe.c

```
Include header files
Create a structure for the universe
Initialize a universe with the pointer function, and take the malloc of the universe
Initialize the matrix as a boolean function and take the calloc of the rows
Create a for loop that checks if the rows are less than r, then initialize the matrix array
Make a pointer from the universe to toroidal equal to toroidal
Make a pointer from the universe to grid equal to the matrix
Return universe

Create a void universe delete function
Set a for loop that checks if the universe rows are less than the integer of r
If so, then free the values of the universe and point the universe towards the grid of r
Free the grid from the pointer of the universe
Free the universe

Create an int function that sets the rows in the universe
Return the pointer of the universe to rows

Create an int function that sets the columns in the universe
Return the pointer of the universe to columns

Create a void function that sets the live cells in the universe
Create an array to print the live cells in the array they are in
Return true if the universe points to the grid of r and c

Create a void function that sets the dead cells in the universe
```

Create an array to print the dead cells in the array they are in

Return false if the universe points to the grid of r and c

Create a boolean function that populates the universe

Initialize a variable for the value of rows

Initialize a variable for the value of columns

Create a while loop that scans through a file and returns the rows and columns in integers

Call universe live cell

If the integer row and columns are less than 0 and the universe rows and columns are greater than the integer, then return false

Return true otherwise

Create a function that creates the census in the universe

Initialize a variable for count

Create 4 integers to check the surrounding neighbors in a toroidal universe

Create 4 integers to check the surrounding neighbors in a flat universe

Sort the toroidal neighbors into rows

Sort the toroidal neighbors into cols

Sort the neighbors into rows

Sort the neighbors into cols

Create a for loop that iterates through both arrays, and if true then add to the counter

If false, get the dead cell

Return the counter outside the for loop

Create a function that prints the universe based on the universe and the file

Create a for loop that starts at an integer 0 and keeps adding 1 if the rows are greater than the integer

Create a for loop that starts at another integer 0 and keeps adding 1 if the cols are greater than the integer

Make an if statement that checks if the cell acquired from the universe is true, then print on the outfile that the cell is alive

Else, print the cell is dead

Print a new line on the outfile

Universe.h

Include header files

Create a structure for the universe

Implement a function to create the universe

Create a function to delete the universe

Create a function that sets the rows and cols of the universe

Create a function that contains the live cells and dead cells in the universe

Create a bool function that gets the live cells in the universe

Create a bool function that populates the universe

Create a function that checks the census of the universe

Create a function that prints the universe

life.c

Include header files

Create a usage function that contains all the options for the user to select when running the program

Create a swap function that can swap the values of universeA with universeB using a temp function.

Create a main function that takes in argc and argv as parameters

Initialize variables for a number of generations, input, output, integer of the row, integer of the col, and opt

Set two boolean variables to false: 1 for the toroidal universe and another for the flat universe

Create a while loop that checks getopt for each case in the options a user can select

For input and output in the cases, set those equal to optarg

Initialize a file with a pointer that can open the input file and read it

Scan the file and store the integer rows and columns

Create two universes with a pointer function

Populate universe with a file

Close the file

Initialize the screen

Hide the cursor

Create a for loop that iterates through count and adds to it as the number of gens are greater than the count

If the ncurses are being displayed, clear the window

Create a for loop that loops through the universeA rows and adds to it based on the count

Create a for loop that loops through the universeA cols and adds to it based on the count

```
If the cell acquired from Universe A is valid, print out a live cell
Else print out a dead cell
Refresh window
Add a delay so output can be readable
Create a for loop to iterate through universe A rows and check the 2nd condition in life rules
Create a for loop that iterates through the columns of universe A and adds 1 to the iterator
Make an if statement that checks if the cell is alive
If the cell is alive and has 2 or 3 neighbors, then print live cell
        Else, print dead cell
If the cell is not alive and has 3 neighbors, then print live cell
        Else price dead cell
Else, print dead cell
Swap the two universes if there is a toroidal universe
Close the screen
Make a second file and open it with an input file and write on it
Print out the universe for universeA using the second file
Closeout the second file
Return 0
```

Credit:

- I attended Eugene's section on 1/28/21, which helped give me general guidance on how to approach this lab, as well as sketch out how the program runs.
- I used my LFLAGS implementation from Omar in discord.
- I used the pseudocode from the asgn4 documentation
- I also used the asgn4 documentation and reviewed the pseudocode provided by Elmer in discord.