Version 1.0

<u>Section 0: universe.c</u>

Universe.c

- Create the Universe struct with the necessary characteristics
  - rows,cols,**grid,toroidal
- Construct a new Universe: Universe *uv_create(rows,cols,toroidal)
  - Create a pointer to a universe, which is an array of length 1 with a size of Universe bytes
  - Set rows to rows, cols to cols, and toroidal to toroidal
  - Then for the grid, we want to create a 2d matrix of bools, with the rows holding bool values
  - The column arrays should hold the bool values
  - Return pointer to allocated memory
- Free allocated memory:
  - Like asignment3.pdf says, it's good practice to free from the inside out, so we first need to free the column values for each row, then the rows, then u itself
- To get the number of rows in a universe, we can return u->rows
- To get the number of cols in a universe, we can return u->cols
- In order to change the state of a cell we can manipulate a cells status by setting u->grid[c][r] to true
  - Do this only if input is within : 0 <= r <= u->rows and 0 <= c <= u->cols
- To return the value of a cell we use the same method for accessing a cell as before and return the state of the cell, once again bounds taken into consideration
- To populate the Universe, read all but the first row from a the supplied files and add all the correct rows to the Universe, and return False if any number inputs are invalid
- To take a census of the Universe, we can check the valid neighbors depending of whether its toroidal or not
- To print the universe:
  - Iterate through the universe and print “o” if the cell is true and “.” if false

Section 1: life.c

- First initialize all useful members from the command line arguments for silenced, gen, input file, etc…
- Remember to add in error handling for all possible error entries whether that be invalid command like args or out of bounds coordinates from the inputfile
- After reading and interpreting the command line, scan the input file for the number of rows and columns for Universe A
- Create Universe A and B of size input_rows and input_cols
- Populate the universe, however if uv_populate returns false then send out an error relaying which coordinates are bad
- Depending on whether ncurses is silenced or not, initscr,loop through the generations universe and mvprintw the correct output,refresh and then sleep.
- Like asgn3 specifies, set the corresponding cell in universe B to the correct state depending on its census
- After each generation swap the universes utilizing a pointer to a universe called temp, and continue through the generation loop.
- When all generations are done, print the final generation and free all allocated memory using uv_delete.

Section 2: Purpose

- This lab was helpful for understanding dynamic memory allocation which should often reflect most real world cases where static data isnt always the method of approach.
- Only in some cases can data be known at the beginning, enough so to create or initialize an array of a certain size seems basic and I enjoyed this lab's theme.
- I’ve taken a python course and obviously structures stand out as similar to Objects and in that they both are typedefs and contain tags of differing types, and, dare I say it, “methods”.

That's only for python of course, in C we create functions with structures as arguments.