Assignment 4 DESIGN.pdf

Ning Jiang

**Description of Program:** For this assignment, we mainly need to creat a ADT(abstract data type) called Universe using struct in universe.c and life.c contains main() and may contain any other functions necessary to complete the implementation of the Game of Life.

**Files to be included in directory "asgn4":**
1. universe.c implements the Universe ADT.
2. universe.h specifies the interface to the Universe ADT. This file is provided and may not be
modified.
3. life.c contains main() and may contain any other functions necessary to complete the implementation of the Game of Life.
4. Makefile
5. README.md
6. DESIGN.pdf

**Structure & explanation:**

- The universe will be abstracted as a struct called Universe.An instance of a Universe must contain the following fields: rows, cols, and a 2-D boolean grid,
  grid.

- allocating a matrix of uint32_ts // do it like Page3 in asgn4

- void uv_delete(Universe *u):

Free everything. In the case of multilevel data structures such as a Universe, we must free the inside first, and then free the outside.

- uint32_t uv_rows(Universe *u):
  Return rows

- uint32_t uv_cols(Universe *u):
  Return cols

- void uv_live_cell(Universe *u, uint32_t r, uint32_t c):

marks the cell at row r and column c as live

- void uv_dead_cell(Universe *u, uint32_t r, uint32_t c):

marks the cell at row r and column c as dead

- bool uv_get_cell(Universe *u, uint32_t r, uint32_t c)

returns the value of the cell at row r and column c

- bool uv_populate(Universe *u, FILE *infile):

use fscanf to scan the file to get the coordinate of live cells.

```
   if the live cell coordinate out of grid
             return false
   Else:
             Set the cell to live
   Return true
```

- uint32_t uv_census(Universe *u, uint32_t r, uint32_t c):

```
    reset the r and c to signed
    for(i in range(r - 1, r1):
       For (j in range(c - 1, c1):
       If the point is (r,c) itself instead of its neighbors
                  continue;
      else if (grid is (toroidal) and (i or j is out of the grid)):
            Do modular arithmetic
            Check if grid[h][g] == 1:
            If true:
               Count = count + 1
            else if (grid is (not toroidal) and (i or j is out of
              the grid)):
               Continue to find the next point.
            Else:
               Count = count + 1
Return count
```

- void uv_print(struct Universe *u, FILE *outfile):

```
      for(i in range(0, rows):
         if the point(i,j) in the grid is True:
            Put 'o' in outfile
         Else:
            Put '.' in outfile
       Put '\n' in outfile
   // give the output to the file
```

- EXTRA FILE TO TEST uv_sensus:

It is not the requirement for this assignment but I think it’s helpful because uv_sensus is the point of this whole assignment.

It is important to make sure that it’s correct.

- Life.c

```
#define DELAY 50000
#define OPTION "tsn:i:o:"

int main(int argc, char **argv){
    uint32_t rows, cols;
    int generations = 0;
    FILE *input = stdin;
    FILE *output = stdout;
    bool ncurses = true;
    bool toroidal = false;
    int opt = 0;
    while ((opt = getopt(argc, argv, OPTION)) != -1)
        switch (opt)
            case 't': toroidal = true; break;
            case 's': ncurses = false; break;
            case 'n': generations = (uint32_t) strtoul(optarg, NULL,
                                    10); break;
            case 'i': input = fopen(optarg, "r"); break; // fopen
            case 'o': output = fopen(optarg, "w"); break;
```

read the rows and cols of the grid by scaning the input file.
Create two universes using the dimension.
Populate universe Ausing uv_populate() with the remainder of the input
Setup the ncurses screen
// from asgn4.pdf
For each generation up to the set number of generations:
(a) If ncurses isn’t silenced by the -s option, clear the screen, display universe A, refresh the screen, then sleep for 50000 microseconds.
(b) Perform one generation. This means taking a census of each cell in universe A and either setting or clearing the corresponding cell in universe B, based off the 3 rules discussed in §2.
(c) Swap the universes. Think of universe A as the current state of the universe and universe B as the next state of the universe. To update the universe then, we simply have to swap A and B. Hint: swapping pointers is much like swapping integers.
7. Close the screen with endwin().
8. Output universe A to the specified file using uv_print(). This is what you will be graded on. We will know if you properly evolved your universe for the set number

of generations by comparing
your output to that of the supplied program.

Notes:
1. Remember to use malloc to allocate a memory to u.
Universe *u = (Universe *) malloc(sizeof(Universe));
2. Use -> pointer to point to each elements in Universe. Like
u->rows = rows;
u->cols = cols;
u->toroidal = toroidal;
3. use modular arithmetic for toroidal grid.
 h = (i + (rows)) % (rows);
 g = (j + (cols)) % (cols);
3. Remember to free everything.
4. In ncurses, instead of using printf, we use printw

**Pseudocode:**
Universe.c:

- Universe *uv_create(uint32_t rows, uint32_t cols, bool toroidal)

allocating a matrix of uint32_ts // do it like Page3 in asgn4

- void uv_delete(Universe *u):

```
for (r in range(0, u -> rows):
         free(grid[r])
free(grid)
free(u)
```

- uint32_t uv_rows(Universe *u):
  return(rows)

- uint32_t uv_cols(Universe *u):
  return(cols)

- void uv_live_cell(Universe *u, uint32_t r, uint32_t c):
  grid[r][c] = true

- void uv_dead_cell(Universe *u, uint32_t r, uint32_t c):

```
grid[r][c] = false
```

- ```
  bool uv_get_cell(Universe *u, uint32_t r, uint32_t c)
  return grid[r][c]
  ```

- ```
  bool uv_populate(Universe *u, FILE *infile): while
  (fscanf(infile,"%d%d", &r, &c) != EOF)
      if ((r < 0 or c < 0) or (r >= rows or c >= cols):
            return false
       Else:
            grid[r][c] = true
  Return true
   uint32_t uv_census(Universe *u, uint32_t r, uint32_t c):
   for(i in range(r - 1, r):
      For (j in range(c - 1, c):
            if(i == r and j == c):
                 continue;
            if (grid is (toroidal) and (i or j is out of the
        grid)):
             h = (i + (rows)) % (rows);
             g = (j + (cols)) % (cols);
            Check if grid[h][g] == 1:
            If true:
               Count = count + 1
          else if (grid is (not toroidal) and (i or j is outofthe
            grid)):
                    Continue to find the next point.
            Else:
               Count = count + 1
  Return count
  ```

- ```
  void uv_print(struct Universe *u, FILE *outfile):
      For in in range(0, rows):
        if grid[i][j] = True:
           fputc('o', outfile)
        Else:
           fputc('.', outfile)
      fputc('\n', outfile)
  // give the output to the file
  ```
  ```

Life.c:
Pseudocode for generation:

```
    While generations:
       If ncurses:
          For i in range(0, rows):
             For j in range(0, cols):
                if uv_get_cell(A, i, j):
                   Print('o')
                Else:
                   Print('w')
          Print('\n')
   refresh();
   usleep(DELAY);

   Perform one generation:
for(r in range(0, rows):
   for(c in range(0, cols):
      int count = uv_census(A, r, c)
      bool state = uv_get_cell(A, r, c)
      if(state):
         if((count == 2) or (count == 3)):
            uv_live_cell(B, r, c)
         Else:
            uv_dead_cell(B, r, c)
      Else:
         if(count == 3):
            uv_live_cell(B, r, c)
         Else:
            uv_dead_cell(B, r, c)
```

**Error Handing:**

1. I use a lot of time to find the error in uv_census. After I write my own testig file, I find out that since r is unsigned integer, when the testing point is (0,0), we want to get its neighbour by using r - 1 which is invalid. As a result, I use another signed integer variable r1 = r and do the same to c: int64_t c1 = c.
2. I my fscanf don't work and cause: Segmentation fault (core

dumped). I set up valgrind to find what’s going on. I found that the reason why this error appears is because I was scan the char instead of FILE.

Credit:
1. I learned how to make an ADT by using struct.
2. I learned how to use ncurses.
3. I learn how to do a terminal grid.
4. I learned how to debug.
5. I learned how to use fscanf.