Name \_\_\_\_Zhou, Juliann\_\_\_\_\_\_\_ Date: \_\_\_\_May 2, 2022\_\_\_\_\_\_\_\_ (last name, first name)

NYU ID: \_\_kyz224\_\_\_ Section: \_\_\_007\_\_\_

**Assignment 2**

**Total in points** (100 points total): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Professor’s Comments:**

**Assignment 2 Report**

In this assignment, we are to simulate the game of life with four rules using 5\*5 matrix:

1. A live cell with fewer than two live neighbors dies.  
2. A live cell with more than three live neighbors also dies.  
3. A live cell with exactly two or three live neighbors lives.  
4. A dead cell with exactly three live neighbors becomes alive.

First, we create an input file which we input a 5\*5 matrix as the initial state.

Rectangle

Description automatically generated with low confidence

Then we read the text file and store the matrix in as a 2-d integer array with the following code:

char c;  
int matrix[5][5];  
  
for(int i =0; i < 5; i++)  
{  
 for(int j=0; j<5; j++)  
 {  
 fscanf(file, " %c", &c);  
 matrix[i][j] = c - '0';  
  
 }  
  
}  
fclose(file);

Next, we implement the function of game of life. We first copy the previous state into another 2-d array using memcpy for future reference.

memcpy(grid, board, sizeof(grid));

We use another 2-d array to store the possible moves of (i, j) from the current cell, which we will use to check the neighbors of the current cell.

int moves[8][2] = {{-1, 1},  
 {-1, 0},  
 {-1, -1},  
 {0, -1},  
 {0, 1},  
 {1, 0},  
 {1, -1},  
 {1, 1}};

Then with the nested for-loops, we iterate through each cell to count the total number of neighbors and store the result in the variable *total*.

for (int i = 0; i < m; i++) {  
 for (int j = 0; j < n; j++) {  
 int total = 0;  
 for (int k = 0; k < dm; k++) {  
 int cr = i + moves[k][0];  
 int cc = j + moves[k][1];  
 if (cr >= 0 && cr < m && cc >= 0 && cc < n) {  
 total += grid[cr][cc];  
 }  
 }

Since we have the total number of neighbors for each cell, we only need to check the four conditionals to determine the next state for the current cell. If it has fewer neighbor than 2 or more neighbors than 3, we set the cell to 0. If it has exactly 3 neighbors, then it will live on to the next state or generate a new cell, so we set the cell to 1. Else, the cell remains unchanged, so we set the value to the one from the initial state, which we can retrieve from grid.

if (total < 2 || total > 3) {  
 board[i][j] = 0;  
} else if (total == 3) {  
 board[i][j] = 1;  
} else {  
 board[i][j] = grid[i][j];  
}

In main, we read the number of generations and the input file name from command and store them in variable n and mapping respectively. Then, we use a while loop with conditional n>0 to generate a new state and decrement n. We use a pointer as the passing parameter and return value for the function game\_of\_life to avoid multiple copies of the matrix, so the pointer will always point to the same location in memory where the matrix is altered. Using nested for-loops, we access the state of each cell in the 2-d array with pointer \*(ptr + (i \* 5) + j) and update the matrix.

while (n > 0){  
 int\* ptr = game\_of\_life(matrix);  
  
 for(int i =0; i < 5; i++)  
 {  
 for(int j=0; j<5; j++)  
 {  
 matrix[i][j] = \*(ptr + (i \* 5) + j);  
 }  
  
 }  
 n--;  
}

Finally, we print and store the final state of the matrix in a text file.

FILE\*pfile=NULL;  
pfile = fopen("output.txt", "w");  
for(int k =0; k < 5; k++)  
{  
 for(int l=0; l<5; l++)  
 {  
 printf("%d", matrix[k][l]);  
 fprintf(pfile, "%d ", matrix[k][l]);  
 }  
 printf("\n");  
 fprintf(pfile, "\n");  
}  
fclose(pfile);

We check the result against refautomata and find that they are correct for 1 and multiple generations.

Refautomata (1 generation):

Text

Description automatically generated with medium confidence

My result (1 generation):

A picture containing rectangle

Description automatically generated

Refautomata (7 generations)

A picture containing graphical user interface

Description automatically generated

My result (7 generations):

Text

Description automatically generated