



Digital Design Verification

FYP/Internship Program

Lab #04

Game of

Life

Name: Abdul Hadi Afzal

Date: 11-March-2025

Contents

TASK1:	3
TASK2:	8
Errors Faced:.....	14
Conclusion:.....	14

TASK1:

CODE: lifegame.h

```
#ifndef LIFEGAME_H_
#define LIFEGAME_H_

/*****
 * ✎ Author: *** Abdul Hadi Afzal ***
 * -= Task: Define constants and function prototypes for Conway's Game
   of Life -=
 *****/

// 🌟 State Constants 🌟
#define DEAD 0 // 💀 Dead Cell
#define ALIVE 1 // ❤️ Alive Cell

// Function Prototypes //
void initialize_world(void);
int get_world_width(void);
int get_world_height(void);
int get_cell_state(int x, int y);
void set_cell_state(int x, int y, int state);
void finalize_evolution(void);
void output_world(void);

#endif /* LIFEGAME_H_ */
```

lifegame.c

```
#include <stdio.h>
#include <stdlib.h>
#include "lifegame.h"

/*****
 * ✎ Author: *** Abdul Hadi Afzal ***
 * -= Task: Implementation of Conway's Game of Life -=
 *****/

#define WORLDWIDTH 39
#define WORLDHEIGHT 20

#define CHAR_ALIVE '*'
#define CHAR_DEAD ' '

static int world[WORLDWIDTH][WORLDHEIGHT];
static int nextstates[WORLDWIDTH][WORLDHEIGHT];
```

```

void initialize_world(void) {
    int i, j;

    for (i = 0; i < WORLDWIDTH; i++)
        for (j = 0; j < WORLDHEIGHT; j++)
            world[i][j] = nextstates[i][j] = DEAD;

    /* Pattern: "Glider" */
    world[1][2] = ALIVE;
    world[3][1] = ALIVE;
    world[3][2] = ALIVE;
    world[3][3] = ALIVE;
    world[2][3] = ALIVE;
}

int get_world_width(void) { return WORLDWIDTH; }
int get_world_height(void) { return WORLDHEIGHT; }

int get_cell_state(int x, int y) {
    if (x < 0 || x >= WORLDWIDTH || y < 0 || y >= WORLDHEIGHT)
        return DEAD;
    return world[x][y];
}

void set_cell_state(int x, int y, int state) {
    if (x < 0 || x >= WORLDWIDTH || y < 0 || y >= WORLDHEIGHT) {
        fprintf(stderr, "Error: coordinates (%d,%d) are invalid.\n",
            x, y);
        abort();
    }
    nextstates[x][y] = state;
}

void finalize_evolution(void) {
    int x, y;
    for (x = 0; x < WORLDWIDTH; x++) {
        for (y = 0; y < WORLDHEIGHT; y++) {
            world[x][y] = nextstates[x][y];
            nextstates[x][y] = DEAD;
        }
    }
}

void output_world(void) {
    char worldstr[2 * WORLDWIDTH + 2];
    int i, j;

    worldstr[2 * WORLDWIDTH + 1] = '\\0';
    worldstr[0] = '+';
    for (i = 1; i < 2 * WORLDWIDTH; i++)
        worldstr[i] = '-';
    worldstr[2 * WORLDWIDTH] = '+';
}

```

```

        puts(worldstr);

    for (i = 0; i <= 2 * WORLDWIDTH; i += 2)
        worldstr[i] = '|';

    for (i = 0; i < WORLDHEIGHT; i++) {
        for (j = 0; j < WORLDWIDTH; j++)
            worldstr[2 * j + 1] = world[j][i] == ALIVE ? CHAR_ALIVE :
                CHAR_DEAD;
        puts(worldstr);
    }


    worldstr[0] = '+';
    for (i = 1; i < 2 * WORLDWIDTH; i++)
        worldstr[i] = '-';
    worldstr[2 * WORLDWIDTH] = '+';
    puts(worldstr);
}

```

lab1a.c

```

#include "lifegame.h"
#include <stdio.h>

/*****
 *  Author: *** Abdul Hadi Afzal ***
 * == Task: Main program for Conway's Game of Life ==
 *****/

void next_generation(void);
int get_next_state(int x, int y);
int num_neighbors(int x, int y);

int main(void) {
    initialize_world();

    printf("Generation 0:\n");
    output_world();

    next_generation();
    printf("Generation 1:\n");
    output_world();

    next_generation();
    printf("Generation 2:\n");
    output_world();

    return 0;
}

void next_generation(void) {

```

```

        int x, y;
        int width = get_world_width();
        int height = get_world_height();

        for (x = 0; x < width; x++) {
            for (y = 0; y < height; y++) {
                set_cell_state(x, y, get_next_state(x, y));
            }
        }

        finalize_evolution();
    }

    int get_next_state(int x, int y) {
        int live_neighbors = num_neighbors(x, y);
        int current_state = get_cell_state(x, y);

        if (current_state == ALIVE) {
            if (live_neighbors < 2 || live_neighbors > 3) {
                return DEAD;
            } else {
                return ALIVE;
            }
        } else {
            if (live_neighbors == 3) {
                return ALIVE;
            } else {
                return DEAD;
            }
        }
    }

    int num_neighbors(int x, int y) {
        int dx, dy;
        int count = 0;

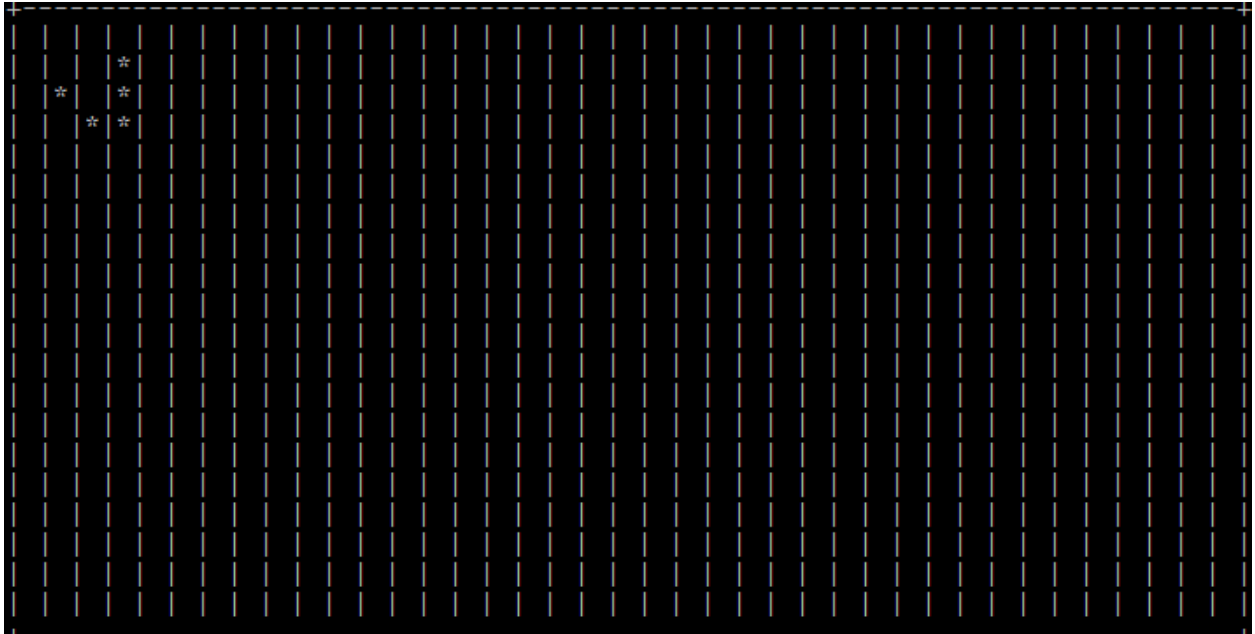
        /* Count live neighbors in the 8 surrounding cells */
        for (dx = -1; dx <= 1; dx++) {
            for (dy = -1; dy <= 1; dy++) {
                if (dx == 0 && dy == 0) continue;
                if (get_cell_state(x + dx, y + dy) == ALIVE) {
                    count++;
                }
            }
        }

        return count;
    }
}

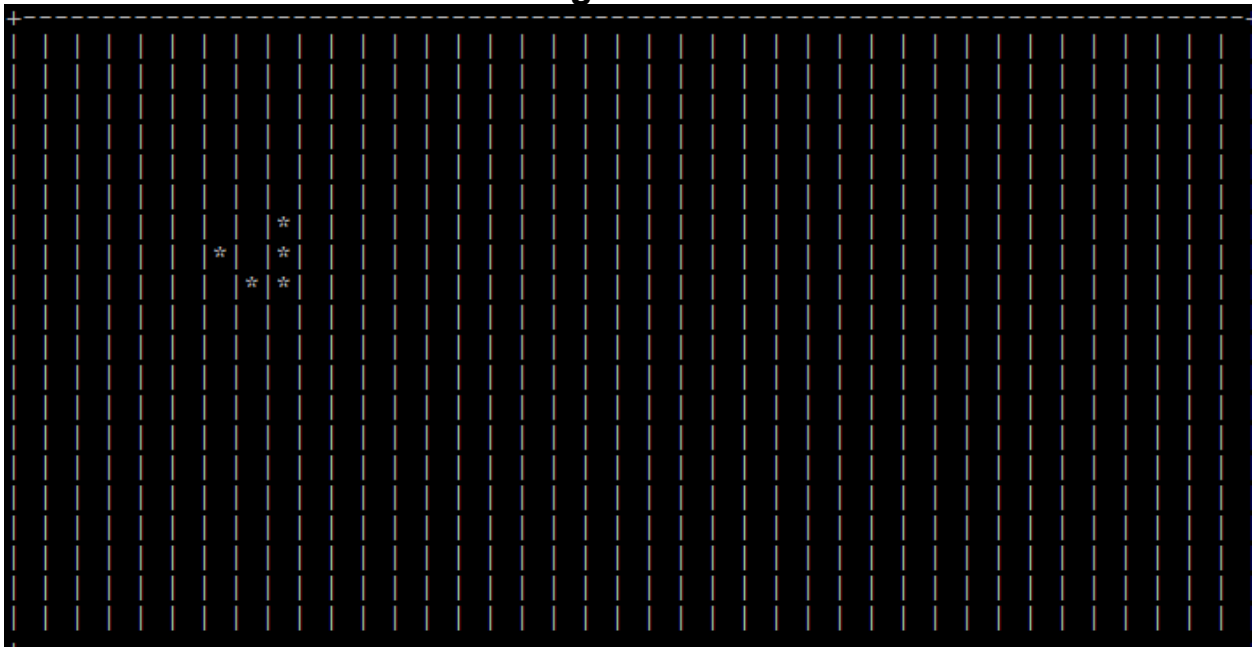
```

SS:

1st Generation

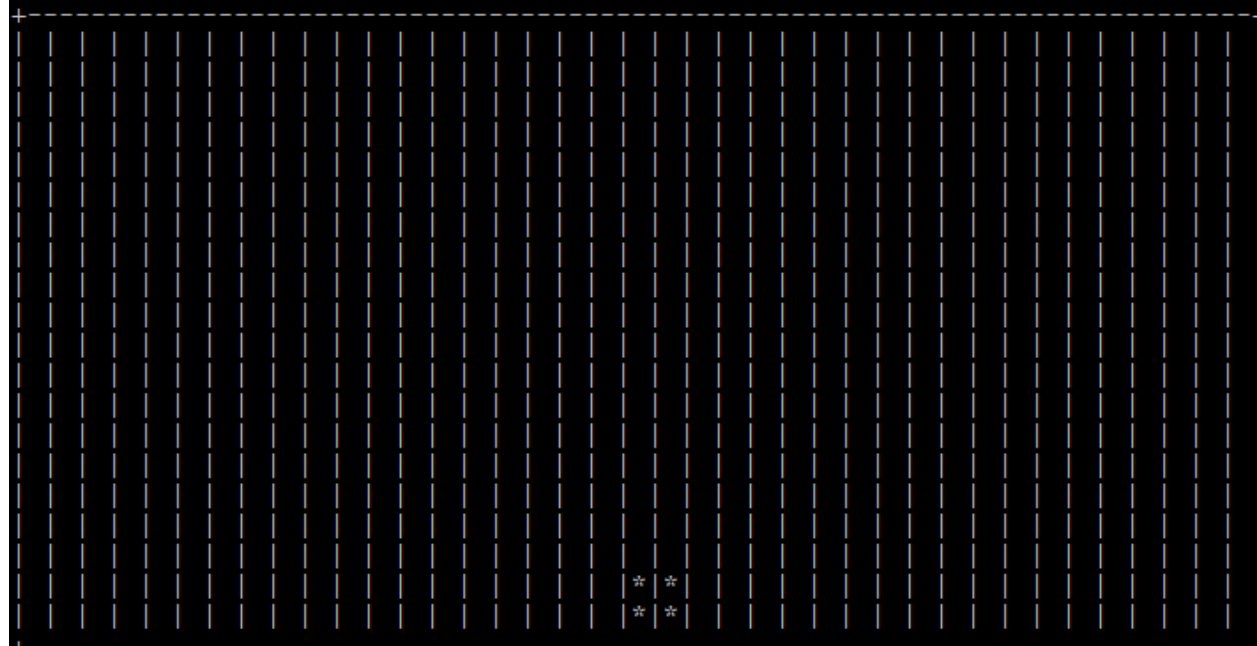


20th generation




Final Generation

Next Generation:



TASK2:

CODE: Lifegame.h

```
/* *****  
 *  Author: Abdul Hadi Afzal  
 * File: lifegame.h  
 * Task: Header file for Conway's Game of Life  
 *****  
 */  
  
#ifndef LIFEGAME_H_  
#define LIFEGAME_H_  
  
#define DEAD 0  
#define ALIVE 1  
  
#define WORLDWIDTH 39  
#define WORLDHEIGHT 20  
  
int get_next_state(int x, int y);  
void initialize_world(void);  
void initialize_world_from_file(const char *filename);  
void save_world_to_file(const char *filename);  
int get_world_width(void);  
int get_world_height(void);
```

```


        int get_cell_state(int x, int y);
        void set_cell_state(int x, int y, int state);
        void finalize_evolution(void);
        void output_world(void);

        #endif /* LIFEGAME_H */

```

lifegame.c

```

/*****
 *  Author: Abdul Hadi Afzal
 * File: lifegame.c
 * Task: Game logic for Conway's Game of Life
 *****/

/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lifegame.h"

#define MAX_LINE_LENGTH 256
#define CHAR_ALIVE '*'
#define CHAR_DEAD ' '

static int world[WORLDWIDTH][WORLDHEIGHT];
static int nextstates[WORLDWIDTH][WORLDHEIGHT];

void next_generation(void) {
    int width = get_world_width();
    int height = get_world_height();

    for (int x = 0; x < width; x++) {
        for (int y = 0; y < height; y++) {
            nextstates[x][y] = get_next_state(x, y);
        }
    }

    finalize_evolution();
}

void initialize_world_from_file(const char *filename) {
    FILE *file = fopen(filename, "r");
    if (!file) {
        perror("Error opening file");
        exit(EXIT_FAILURE);
    }

```

```

    }

    char line[MAX_LINE_LENGTH];
    int y = 0;

    while (fgets(line, MAX_LINE_LENGTH, file) && y <
           WORLDHEIGHT) {
        int x = 0;
        for (int i = 0; line[i] != '\n' && line[i] != '\0'; i++)
        {
            if (line[i] == '*') {
                set_cell_state(x, y, ALIVE);
            }
            x++;
        }
        y++;
    }

    fclose(file);
}

void save_world_to_file(const char *filename) {
    FILE *file = fopen(filename, "w");
    if (!file) {
        perror("Error creating file");
        exit(EXIT_FAILURE);
    }

    for (int y = 0; y < get_world_height(); y++) {
        for (int x = 0; x < get_world_width(); x++) {
            fputc(get_cell_state(x, y) == ALIVE ? '*' : ' ',
                  file);
        }
        fputc('\n', file);
    }

    fclose(file);
}

void initialize_world(void) {
    for (int i = 0; i < WORLDWIDTH; i++)
        for (int j = 0; j < WORLDHEIGHT; j++)
            world[i][j] = nextstates[i][j] = DEAD;

    world[2][4] = ALIVE;
    world[4][4] = ALIVE;
    world[5][5] = ALIVE;

```

```

        world[2][6] = ALIVE;
        world[5][6] = ALIVE;
        world[3][7] = ALIVE;
        world[4][7] = ALIVE;
        world[5][7] = ALIVE;
        world[6][7] = ALIVE;
    }

    int get_world_width(void) {
        return WORLDWIDTH;
    }

    int get_world_height(void) {
        return WORLDHEIGHT;
    }

    int get_cell_state(int x, int y) {
        return (x >= 0 && x < WORLDWIDTH && y >= 0 && y <
            WORLDHEIGHT) ? world[x][y] : DEAD;
    }

    void set_cell_state(int x, int y, int state) {
        if (x >= 0 && x < WORLDWIDTH && y >= 0 && y < WORLDHEIGHT) {
            world[x][y] = state;
            nextstates[x][y] = state;
        }
    }

    void finalize_evolution(void) {
        for (int x = 0; x < WORLDWIDTH; x++) {
            for (int y = 0; y < WORLDHEIGHT; y++) {
                world[x][y] = nextstates[x][y];
                nextstates[x][y] = DEAD;
            }
        }
    }

    void output_world(void) {
        printf("+");
        for (int i = 0; i < 2 * WORLDWIDTH; i++) printf("-");
        printf("+\n");

        for (int y = 0; y < WORLDHEIGHT; y++) {
            printf("|");
            for (int x = 0; x < WORLDWIDTH; x++) {
                printf("%c", world[x][y] == ALIVE ? '*' : ' ');
            }
        }
    }

```

```


        printf("|\\n");
    }

    printf("+");
    for (int i = 0; i < 2 * WORLDWIDTH; i++) printf("-");
    printf("+\\n");
}

```

lab1b.c

```

/*****
 *  Author: Abdul Hadi Afzal
 * File: lab1b.c
 * Task: Main program for Conway's Game of Life
 *****/

#include "lifegame.h"
#include <stdio.h>
#include <stdlib.h>

void next_generation(void);
int get_next_state(int x, int y);
int num_neighbors(int x, int y);

int main(int argc, char *argv[]) {
    if (argc > 1) {
        initialize_world_from_file(argv[1]);
    } else {
        initialize_world();
    }

    printf("Initial Generation:\\n");
    output_world();

    for (int i = 0; i < 50; i++) {
        next_generation();
    }

    printf("Generation %d:\\n", 50);
    output_world();

    save_world_to_file("world.txt");
    return 0;
}

int get_next_state(int x, int y) {

```

```

int live_neighbors = num_neighbors(x, y);
int current_state = get_cell_state(x, y);

    if (current_state == ALIVE) {
return (live_neighbors < 2 || live_neighbors > 3) ? DEAD
        : ALIVE;
    } else {
return (live_neighbors == 3) ? ALIVE : DEAD;
    }
}

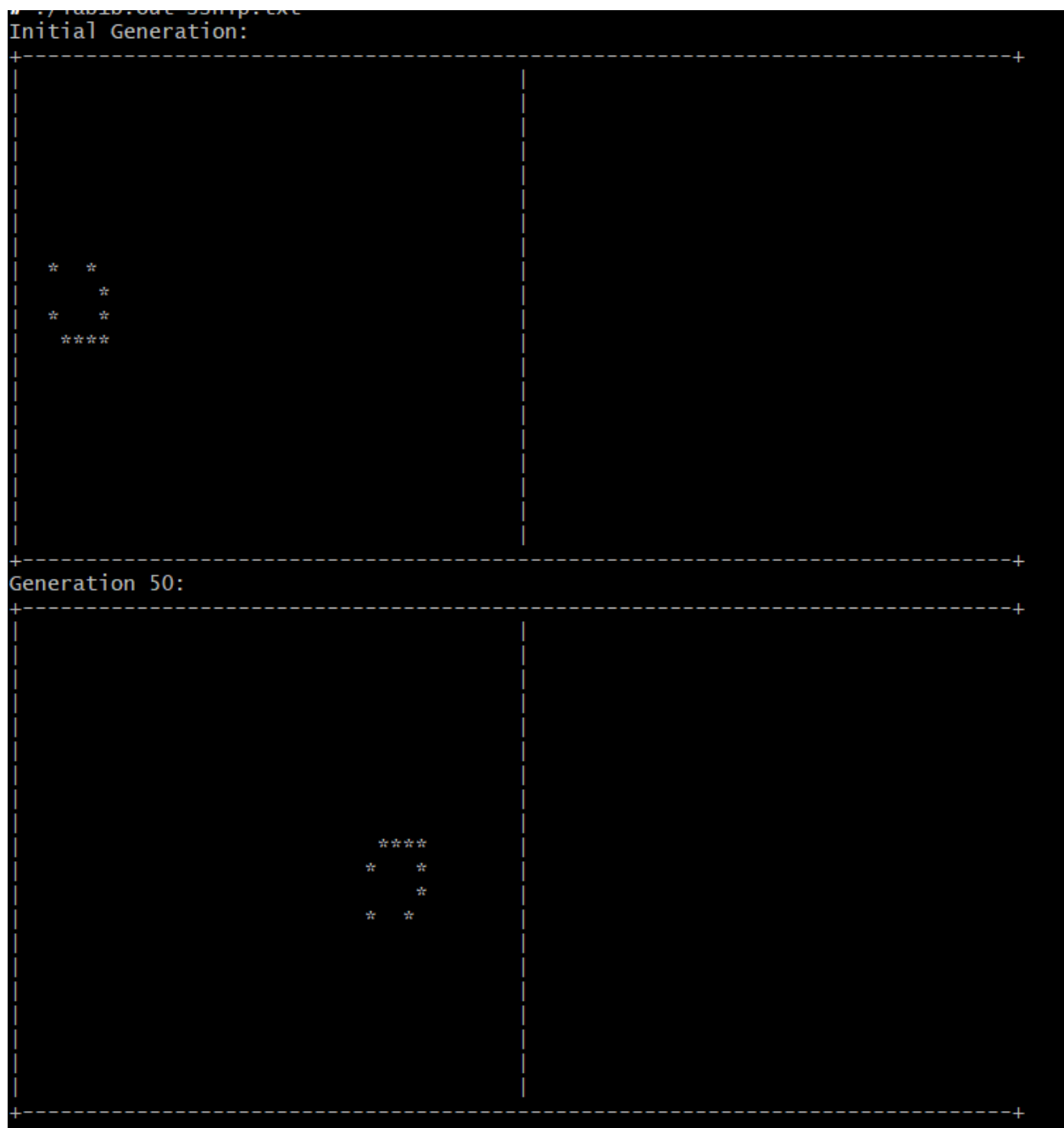
int num_neighbors(int x, int y) {
    int count = 0;
    for (int dx = -1; dx <= 1; dx++) {
        for (int dy = -1; dy <= 1; dy++) {
            if (dx == 0 && dy == 0) continue;

            int neighbor_x = x + dx;
            int neighbor_y = y + dy;

            if (neighbor_x >= 0 && neighbor_x < WORLDWIDTH &&
                neighbor_y >= 0 && neighbor_y < WORLDHEIGHT) {
                if (get_cell_state(neighbor_x, neighbor_y) ==
                    ALIVE) {
                    count++;
                }
            }
        }
    }
    return count;
}

```

SS:



Errors Faced:

I had issues with reading the input file correctly, which caused blank outputs. Debugging errors like missing functions and incorrect neighbor calculations also affected the simulation. Fixing the logic for cell state updates adding debugging messages helped solve these problems.

Conclusion:

This project successfully implements Conway's Game of Life, reading an initial

pattern from a file and simulating its evolution over multiple generations. The final output is displayed on the console and saved to a file for verification.