

```

1 //
2 // Created by hfwei on 2023/10/19.
3 //
4
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <unistd.h>
8 #include <synchapi.h>
9
10 #define SIZE 6
11 const int board[SIZE][SIZE] = {
12     { 0 },
13     { 0, 1, 1, 0, 0, 0 },
14     { 0, 1, 1, 0, 0, 0 },
15     { 0, 0, 0, 1, 1, 0 },
16     { 0, 0, 0, 1, 1, 0 },
17     { 0 }
18 };
19
20 //const int board[SIZE][SIZE] = {
21 //    [1][1] = 1, [1][2] = 1,
22 //    [2][1] = 1, [2][2] = 1,
23 //    [3][3] = 1, [3][4] = 1,
24 //    [4][3] = 1, [4][4] = 1
25 //};
26
27 int main() {
28     // extended board
29     int old_board[SIZE + 2][SIZE + 2] = { 0 };
30
31     for (int row = 1; row <= SIZE; row++) {
32         for (int col = 1; col <= SIZE; col++) {
33             old_board[row][col] = board[row - 1][col - 1];
34         }
35     }
36
37     // print the original board
38     for (int row = 1; row <= SIZE; row++) {
39         for (int col = 1; col <= SIZE; col++) {
40             printf("%c ", old_board[row][col] ? '*' : ' ');
41         }
42         printf("\n");
43     }
44     system("clear"); // clear the screen/terminal
45
46     int new_board[SIZE + 2][SIZE + 2] = { 0 };
47
48     for (int round = 1; round < 10; round++) {
49         for (int row = 1; row <= SIZE; row++) {
50             for (int col = 1; col <= SIZE; col++) {
51                 // count the number of neighbours of old_board[row][col]
52                 int neighbours =
53                     old_board[row - 1][col - 1] +

```

```

54         old_board[row - 1][col] +
55         old_board[row - 1][col + 1] +
56         old_board[row][col - 1] +
57         old_board[row][col + 1] +
58         old_board[row + 1][col - 1] +
59         old_board[row + 1][col] +
60         old_board[row + 1][col + 1];
61
62     // evaluate the new board
63     if (old_board[row][col]) { // old_board[row][col] is alive
64         new_board[row][col] = (neighbours == 2 || neighbours == 3);
65     } else { // old_board[row][col] is dead
66         new_board[row][col] = (neighbours == 3);
67     }
68 }
69 }
70
71 // print the new board
72 for (int row = 1; row <= SIZE; row++) {
73     for (int col = 1; col <= SIZE; col++) {
74         printf("%c ", new_board[row][col] ? '*' : ' ');
75     }
76     printf("\n");
77 }
78
79 // sleep for a while
80 // Linux: #include <unistd.h>
81 sleep(1);
82 // Windows: #include <windows.h>: Sleep(ms)
83 // Sleep(1000);
84
85 // clear the screen
86 // Linux: #include <stdlib.h>
87 system("clear");
88 // Windows: #include <stdlib.h> system("clr");
89 // system("clr");
90
91 // start the next round
92 for (int row = 1; row <= SIZE; row++) {
93     for (int col = 1; col <= SIZE; col++) {
94         old_board[row][col] = new_board[row][col];
95     }
96 }
97 }
98
99 return 0;
100 }

```

```
1 //
2 // Created by hfwei on 2023/10/19.
3 //
4
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <time.h>
8 #include <unistd.h>
9
10 // Define grid dimensions
11 #define ROWS 20
12 #define COLS 40
13
14 // Function to initialize the grid randomly
15 void initializeGrid(int grid[ROWS][COLS]) {
16     for (int i = 0; i < ROWS; i++) {
17         for (int j = 0; j < COLS; j++) {
18             grid[i][j] = rand() % 2; // 0 (dead) or 1 (alive)
19         }
20     }
21 }
22
23 // Function to print the grid
24 void printGrid(int grid[ROWS][COLS]) {
25     for (int i = 0; i < ROWS; i++) {
26         for (int j = 0; j < COLS; j++) {
27             if (grid[i][j] == 1) {
28                 printf("#"); // Alive cell
29             } else {
30                 printf(" "); // Dead cell
31             }
32         }
33         printf("\n");
34     }
35     printf("\n");
36 }
37
38 // Function to update the grid for the next generation
39 void updateGrid(int grid[ROWS][COLS]) {
40     int newGrid[ROWS][COLS];
41
42     for (int i = 0; i < ROWS; i++) {
43         for (int j = 0; j < COLS; j++) {
44             int neighbors = 0;
45
46             // Count neighbors
47             for (int x = -1; x <= 1; x++) {
48                 for (int y = -1; y <= 1; y++) {
49                     if (x == 0 && y == 0) { continue; } // Skip the current
cell
50                     int newX = i + x;
51                     int newY = j + y;
52
```

```

53         if (newX >= 0 && newX < ROWS && newY >= 0 && newY < COLS) {
54             neighbors += grid[newX][newY];
55         }
56     }
57 }
58
59 // Apply Game of Life rules
60 if (grid[i][j] == 1) {
61     newGrid[i][j] = (neighbors == 2 || neighbors == 3) ? 1 : 0;
62 } else {
63     newGrid[i][j] = (neighbors == 3) ? 1 : 0;
64 }
65 }
66 }
67
68 // Update the grid
69 for (int i = 0; i < ROWS; i++) {
70     for (int j = 0; j < COLS; j++) {
71         grid[i][j] = newGrid[i][j];
72     }
73 }
74 }
75
76 int main() {
77     int grid[ROWS][COLS];
78
79     // Seed the random number generator with the current time
80     srand(time(NULL));
81
82     // Initialize the grid
83     initializeGrid(grid);
84
85     // Number of generations
86     int generations = 50;
87
88     for (int gen = 0; gen < generations; gen++) {
89         system("clear"); // Use "clear" on Unix-based systems (Linux,
macOS)
90         printf("Generation %d:\n", gen);
91         printGrid(grid);
92         updateGrid(grid);
93         sleep(1); // Sleep for 100ms
94     }
95
96     return 0;
97 }

```

```
1 //
2 // Created by hfwei on 2023/10/19.
3 //
4
5 #include <stdio.h>
6
7 #define LEN_L 5
8 #define LEN_R 6
9
10 int L[LEN_L] = { 1, 3, 5, 7, 9 };
11 int R[LEN_R] = { 0, 2, 4, 6, 8, 10 };
12
13 int main(void) {
14     int l = 0;
15     int r = 0;
16
17     while (l < LEN_L && r < LEN_R) {
18         if (L[l] <= R[r]) {
19             printf("%d ", L[l]);
20             l++;
21         } else {
22             printf("%d ", R[r]);
23             r++;
24         }
25     }
26
27     while (r < LEN_R) {
28         printf("%d ", R[r]);
29         r++;
30     }
31
32     while (l < LEN_L) {
33         printf("%d ", L[l]);
34         l++;
35     }
36
37     // l >= LEN_L || r >= LEN_R
38     // if (l >= LEN_L) {
39     //     while (r < LEN_R) {
40     //         printf("%d ", R[r]);
41     //         r++;
42     //     }
43     // }
44
45     // if (r >= LEN_R) {
46     //     while (l < LEN_L) {
47     //         printf("%d ", L[l]);
48     //         l++;
49     //     }
50     // }
51
52     return 0;
53 }
```

```
1 # 4-loops
2
3 - `Alt + 6`: Problems on the status bar
4 - `SonarLint` on the status bar
5
6 ## `game-of-life.c`
7
8 - play with it
9 - [wiki](https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life)
10 - [Demo](https://playgameoflife.com/)
11 - [Gosper_glider_gun](https://playgameoflife.com/lexicon/
    Gosper_glider_gun)
12 - [LifeWiki](https://conwaylife.com/wiki/Main_Page)
13 - [Life Lexicon Home Page](https://conwaylife.com/ref/lexicon/
    lex_home.htm)
14 - 2D-array
15 - initialization (Section 8.2.1)
16     - row-major
17     - row by row
18     - indicator
19 - extension of board
20 - how many boards?
21 - one round
22 - multiple rounds
23 - pause
24 - screen clear
25 - [ ] try a new board?
26 - [Life Lexicon Home Page](https://conwaylife.com/ref/lexicon/
    lex_home.htm)
27
28 # `merge.c`
29
30 - examples
31 - for `merge-sort.c` later
```