

```
1 // Created by hfwei on 2024/10/16.
2
3 #include <stdio.h>
4
5 #define LEN_L 5
6 #define LEN_R 6
7
8 int L[LEN_L] = { 1, 3, 5, 7, 9 };
9 int R[LEN_R] = { 0, 2, 4, 6, 8, 10 };
10
11 int main(void) {
12     // TODO: merge L and R into a sorted array
13
14     return 0;
15 }
```

```
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%
10     27s Game of Life)
11   - [Demo](https://playgameoflife.com/)
12   - [Gosper_glider_gun](https://playgameoflife.com/lexicon
13     /Gosper\_glider\_gun)
14   - [LifeWiki](https://conwaylife.com/wiki/Main\_Page)
15   - [Life Lexicon Home Page](https://conwaylife.com/ref/
16     lexicon/lex\_home.htm)
17 - 2D-array
18   - initialization (Section 8.2.1)
19     - row-major
20     - row by row
21     - indicator
22 - extension of board
23 - how many boards?
24 - one round
25 - multiple rounds
26 - pause
27 - screen clear
28 - [ ] try a new board?
29   - [Life Lexicon Home Page](https://conwaylife.com/ref/
30     lexicon/lex\_home.htm)
31
32 # `merge.c`
33
34 - examples
35 - for `merge-sort.c` later
36
37 # `insertion-sort.c`
38
39 - `for` + `while` version
40 - `for` + `for` version
```

```
1 add_executable(game-of-life game-of-life.c)
2 add_executable(game-of-life-chatgpt game-of-life-chatgpt.c
3 )
4 add_executable(insertion-sort insertion-sort.c)
5 add_executable(insertion-sort-bsearch insertion-sort-
6 bsearch.c)
7
8 add_executable(merge merge.c)
```

```
1 // Created by hfwei on 2024/10/16.
2
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <stdlib.h>
6
7 #define SIZE 6
8
9 const int board[SIZE][SIZE] = {
10     { 0 },
11     { 0, 1, 1, 0, 0, 0 },
12     { 0, 1, 1, 0, 0, 0 },
13     { 0, 0, 0, 1, 1, 0 },
14     { 0, 0, 0, 1, 1, 0 },
15     { 0 }
16 };
17
18 //const int board[SIZE][SIZE] = {
19 //    [1][1] = 1, [1][2] = 1,
20 //    [2][1] = 1, [2][2] = 1,
21 //    [3][3] = 1, [3][4] = 1,
22 //    [4][3] = 1, [4][4] = 1
23 //};
24
25 int main(void) {
26     // TODO: play game-of-life
27
28     return 0;
29 }
```

```
1 // Created by hfwei on 2024/10/16.
2 // Code generated by ChatGPT.
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <time.h>
7
8 #define MAX_LEN 10000
9 #define RANGE 100
10
11 int main(void) {
12     int numbers[MAX_LEN] = { 0 };
13
14     int size = 0;
15     scanf("%d", &size);
16
17     srand(time(NULL));
18     for (int i = 0; i < size; i++) {
19         numbers[i] = rand() % RANGE;
20     }
21
22     // print the original array
23     for (int i = 0; i < size; i++) {
24         printf("%d ", numbers[i]);
25     }
26     printf("\n");
27
28     // TODO: insertion sort
29
30     // print the sorted array
31     for (int i = 0; i < size; i++) {
32         printf("%d ", numbers[i]);
33     }
34     printf("\n");
35
36     return 0;
37 }
```

```
1 // Created by hfwei on 2024/10/16.
2 // Code generated by ChatGPT.
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <time.h>
7 #include <unistd.h>
8
9 // Define grid dimensions
10 #define ROWS 20
11 #define COLS 40
12
13 // Function to initialize the grid randomly
14 void initializeGrid(int grid[ROWS][COLS]) {
15     for (int i = 0; i < ROWS; i++) {
16         for (int j = 0; j < COLS; j++) {
17             grid[i][j] = rand() % 2; // 0 (dead) or 1 (alive)
18         }
19     }
20 }
21
22 // Function to print the grid
23 void printGrid(int grid[ROWS][COLS]) {
24     for (int i = 0; i < ROWS; i++) {
25         for (int j = 0; j < COLS; j++) {
26             if (grid[i][j] == 1) {
27                 printf("#"); // Alive cell
28             } else {
29                 printf(" "); // Dead cell
30             }
31         }
32         printf("\n");
33     }
34     printf("\n");
35 }
36
37 // Function to update the grid for the next generation
38 void updateGrid(int grid[ROWS][COLS]) {
39     int newGrid[ROWS][COLS];
40
41     for (int i = 0; i < ROWS; i++) {
42         for (int j = 0; j < COLS; j++) {
43             int neighbors = 0;
```

```

45     // Count neighbors
46     for (int x = -1; x <= 1; x++) {
47         for (int y = -1; y <= 1; y++) {
48             if (x == 0 && y == 0) { continue; } // Skip the
current cell
49             int newX = i + x;
50             int newY = j + y;
51
52             if (newX >= 0 && newX < ROWS && newY >= 0 &&
newY < COLS) {
53                 neighbors += grid[newX][newY];
54             }
55         }
56     }
57
58     // Apply Game of Life rules
59     if (grid[i][j] == 1) {
60         newGrid[i][j] = (neighbors == 2 || neighbors == 3
) ? 1 : 0;
61     } else {
62         newGrid[i][j] = (neighbors == 3) ? 1 : 0;
63     }
64 }
65 }
66
67 // Update the grid
68 for (int i = 0; i < ROWS; i++) {
69     for (int j = 0; j < COLS; j++) {
70         grid[i][j] = newGrid[i][j];
71     }
72 }
73 }
74
75 int main(void) {
76     int grid[ROWS][COLS];
77
78     // Seed the random number generator with the current
time
79     srand(time(NULL));
80
81     // Initialize the grid
82     initializeGrid(grid);
83
84     // Number of generations

```

```
85  int generations = 50;
86
87  for (int gen = 0; gen < generations; gen++) {
88      system("clear"); // Use "clear" on Unix-based
        systems (Linux, macOS)
89      printf("Generation %d:\n", gen);
90      printGrid(grid);
91      updateGrid(grid);
92      sleep(1); // Sleep for 100ms
93  }
94
95  return 0;
96 }
```



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 #define MAX_LEN 10000
6 #define RANGE 10
7
8 int main() {
9     int numbers[MAX_LEN] = { 0 };
10
11     int size = 0;
12     scanf("%d", &size);
13
14     srand(time(NULL));
15     for (int i = 0; i < size; i++) {
16         numbers[i] = rand() % RANGE;
17     }
18
19     // print the original array
20     for (int i = 0; i < size; i++) {
21         printf("%d ", numbers[i]);
22     }
23     printf("\n");
24
25     // TODO: binary insertion sort
26
27     // Print the sorted array
28     for (int i = 0; i < size; i++) {
29         printf("%d ", numbers[i]);
30     }
31     printf("\n");
32
33     return 0;
34 }
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 #define MAX_LEN 10000
6 #define RANGE 10
7
8 int main() {
9     int numbers[MAX_LEN] = { 0 };
10
11     int size = 0;
12     scanf("%d", &size);
13
14     srand(time(NULL));
15     for (int i = 0; i < size; i++) {
16         numbers[i] = rand() % RANGE;
17     }
18
19     // print the original array
20     for (int i = 0; i < size; i++) {
21         printf("%d ", numbers[i]);
22     }
23     printf("\n");
24
25     // TODO: insertion sort with binary search
26
27     // Print the sorted array
28     for (int i = 0; i < size; i++) {
29         printf("%d ", numbers[i]);
30     }
31     printf("\n");
32
33     return 0;
34 }
```