```c
//
// Created by hfwei on 2023/10/19.
//

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <synchapi.h>

#define SIZE 6
const int board[SIZE][SIZE] = {
    { 0 },
    { 0, 1, 1, 0, 0, 0 },
    { 0, 1, 1, 0, 0, 0 },
    { 0, 0, 0, 1, 1, 0 },
    { 0, 0, 0, 1, 1, 0 },
    { 0 }
};

//const int board[SIZE][SIZE] = {
//    [1][1] = 1, [1][2] = 1,
//    [2][1] = 1, [2][2] = 1,
//    [3][3] = 1, [3][4] = 1,
//    [4][3] = 1, [4][4] = 1
//};

int main() {
  // extended board
  int old_board[SIZE + 2][SIZE + 2] = { 0 };

  for (int row = 1; row <= SIZE; row++) {
    for (int col = 1; col <= SIZE; col++) {
      old_board[row][col] = board[row - 1][col - 1];
    }
  }

  // print the original board
  for (int row = 1; row <= SIZE; row++) {
    for (int col = 1; col <= SIZE; col++) {
      printf("%c ", old_board[row][col] ? '*' : ' ');
    }
    printf("\n");
  }
  system("clear");  // clear the screen/terminal

  int new_board[SIZE + 2][SIZE + 2] = { 0 };

  for (int round = 1; round < 10; round++) {
    for (int row = 1; row <= SIZE; row++) {
      for (int col = 1; col <= SIZE; col++) {
        // count the number of neighbours of old_board[row][col]
        int neighbours =
            old_board[row - 1][col - 1] +
                  old_board[row - 1][col] +
                  old_board[row - 1][col + 1] +
                  old_board[row][col - 1] +
                  old_board[row][col + 1] +
                  old_board[row + 1][col - 1] +
                  old_board[row + 1][col] +
                  old_board[row + 1][col + 1];

        // evaluate the new board
        if (old_board[row][col]) {  // old_board[row][col] is alive
          new_board[row][col] = (neighbours == 2 || neighbours == 3);
        } else {  // old_board[row][col] is dead
          new_board[row][col] = (neighbours == 3);
        }
      }
    }

    // print the new board
    for (int row = 1; row <= SIZE; row++) {
      for (int col = 1; col <= SIZE; col++) {
        printf("%c ", new_board[row][col] ? '*' : ' ');
      }
      printf("\n");
    }

    // sleep for a while
    // Linux: #include <unistd.h>
    sleep(1);
    // Windows: #include <windows.h>: Sleep(ms)
    // Sleep(1000);

    // clear the screen
    // Linux: #include <stdlib.h>
    system("clear");
    // Windows: #include <stdlib.h> system("clr);
    // system("clr");

    // start the next round
    for (int row = 1; row <= SIZE; row++) {
      for (int col = 1; col <= SIZE; col++) {
        old_board[row][col] = new_board[row][col];
      }
    }
  }

  return 0;
}
```

```c
//
// Created by hfwei on 2023/10/19.
//

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

// Define grid dimensions
#define ROWS 20
#define COLS 40

// Function to initialize the grid randomly
void initializeGrid(int grid[ROWS][COLS]) {
  for (int i = 0; i < ROWS; i++) {
    for (int j = 0; j < COLS; j++) {
      grid[i][j] = rand() % 2;  // 0 (dead) or 1 (alive)
    }
  }
}

// Function to print the grid
void printGrid(int grid[ROWS][COLS]) {
  for (int i = 0; i < ROWS; i++) {
    for (int j = 0; j < COLS; j++) {
      if (grid[i][j] == 1) {
        printf("#");  // Alive cell
      } else {
        printf(" ");  // Dead cell
      }
    }
    printf("\n");
  }
  printf("\n");
}

// Function to update the grid for the next generation
void updateGrid(int grid[ROWS][COLS]) {
  int newGrid[ROWS][COLS];

  for (int i = 0; i < ROWS; i++) {
    for (int j = 0; j < COLS; j++) {
      int neighbors = 0;

      // Count neighbors
      for (int x = -1; x <= 1; x++) {
        for (int y = -1; y <= 1; y++) {
          if (x == 0 && y == 0) { continue; }  // Skip the current cell
          int newX = i + x;
          int newY = j + y;

          if (newX >= 0 && newX < ROWS && newY >= 0 && newY < COLS) {
            neighbors += grid[newX][newY];
          }
        }
      }

      // Apply Game of Life rules
      if (grid[i][j] == 1) {
        newGrid[i][j] = (neighbors == 2 || neighbors == 3) ? 1 : 0;
      } else {
        newGrid[i][j] = (neighbors == 3) ? 1 : 0;
      }
    }
  }

  // Update the grid
  for (int i = 0; i < ROWS; i++) {
    for (int j = 0; j < COLS; j++) {
      grid[i][j] = newGrid[i][j];
    }
  }
}

int main() {
  int grid[ROWS][COLS];

  // Seed the random number generator with the current time
  srand(time(NULL));

  // Initialize the grid
  initializeGrid(grid);

  // Number of generations
  int generations = 50;

  for (int gen = 0; gen < generations; gen++) {
    system("clear");  // Use "clear" on Unix-based systems (Linux, macOS)
    printf("Generation %d:\n", gen);
    printGrid(grid);
    updateGrid(grid);
    sleep(1);  // Sleep for 100ms
  }

  return 0;
}
```

```c
//
// Created by hfwei on 2023/10/19.
//

#include <stdio.h>

#define LEN_L 5
#define LEN_R 6

int L[LEN_L] = { 1, 3, 5, 7, 9 };
int R[LEN_R] = { 0, 2, 4, 6, 8, 10 };

int main(void) {
  int l = 0;
  int r = 0;

  while (l < LEN_L && r < LEN_R) {
    if (L[l] <= R[r]) {
      printf("%d ", L[l]);
      l++;
    } else {
      printf("%d ", R[r]);
      r++;
    }
  }

  while (r < LEN_R) {
    printf("%d ", R[r]);
    r++;
  }

  while (l < LEN_L) {
    printf("%d ", L[l]);
    l++;
  }

  // l >= LEN_L || r >= LEN_R
  // if (l >= LEN_L) {
  //   while (r < LEN_R) {
  //     printf("%d ", R[r]);
  //     r++;
  //   }
  // }

  // if (r >= LEN_R) {
  //   while (l < LEN_L) {
  //     printf("%d ", L[l]);
  //     l++;
  //   }
  // }

  return 0;
}
```

```markdown
# 4-loops

- `Alt + 6`: Problems on the status bar
- `SonarLint` on the status bar

## `game-of-life.c`

- play with it
  - [wiki](https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life)
  - [Demo](https://playgameoflife.com/)
  - [Gosper_glider_gun](https://playgameoflife.com/lexicon/Gosper_glider_gun)
  - [LifeWiki](https://conwaylife.com/wiki/Main_Page)
  - [Life Lexicon Home Page](https://conwaylife.com/ref/lexicon/lex_home.htm)
- 2D-array
  - initialization (Section 8.2.1)
    - row-major
    - row by row
    - indicator
- extension of board
- how many boards?
- one round
- multiple rounds
- pause
- screen clear
- [ ] try a new board?
  - [Life Lexicon Home Page](https://conwaylife.com/ref/lexicon/lex_home.htm)

# `merge.c`

- examples
- for `merge-sort.c` later
```