```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)
  - [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

# `bubble-sort.c`

- generating test cases
- timing
- `swapped`
  - TAOCP Vol. 3 (P109)
```

```c
//
// Created by hengxin on 10/19/22.
//

#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() {
  int l = 0;
  int r = 0;

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

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

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

  return 0;
}
```

```c
//
// Created by hengxin on 10/19/22.
// Run it with "Terminal"
//

#include <stdio.h>
#include <stdlib.h>
#include <unistd.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() {
  int old_board[SIZE + 2][SIZE + 2];
  for (int row = 0; row < SIZE + 2; row++) {
    for (int col = 0; col < SIZE + 2; col++) {
      if (row == 0 || row == SIZE + 1 || col == 0 || col == SIZE + 1
) {
        old_board[row][col] = 0;
      } else {
        old_board[row][col] = board[row - 1][col - 1];
      }
    }
  }

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

  // clear the screen
  system("clear");

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

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

    // clear the screen
    // Windows: #include <conio.h> clrscr();
    system("clear");

    // start the next round
    for (int row = 0; row < SIZE + 2; row++) {
      for (int col = 0; col < SIZE + 2; col++) {
        old_board[row][col] = new_board[row][col];
      }
    }
  }

  return 0;
}
```

```c
//
// Created by hengxin on 10/19/22.
//
// For the usage of clock(),
// please refer to https://legacy.cplusplus.com/reference/ctime/clock/.
//

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

#define LEN 1000000
int numbers[LEN] = {0};

int main() {
  srand(time(NULL));
  for (int i = 0; i < LEN; i++) {
    numbers[i] = rand() % LEN;
  }

  // set a clock
  clock_t start = clock();

  bool swapped = true;
  for (int i = 0; i < LEN && swapped; i++) {
    swapped = false;
    for (int j = 0; j < LEN - 1 - i; j++) {
      if (numbers[j] > numbers[j + 1]) {
        int temp = numbers[j];
        numbers[j] = numbers[j + 1];
        numbers[j + 1] = temp;
        swapped = true;
      }
    }
  }

  // record the end clock
  clock_t end = clock();

  for (int i = 0; i < LEN; i++) {
    printf("%d ", numbers[i]);
  }

  long sec = (end - start) / CLOCKS_PER_SEC;
  printf("Sorting %d numbers in %ld seconds.\n", LEN, sec);

  return 0;
}
```