lab11

```
$ gcc life.c main.o
$ ./a.out < pat1.dat
Generation 30
. . O . . . O . . . . . O . . . O . . .
. . O O . . O . . . . . O . . O O . . .
. . . . . . O . . . . . O . . . . . . .
O O . . . . O O . . . O O . . . . O O .
. O . . O . . . . . . . . . O . . O . .
. . O . O . . . . . . . . . O . O . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . O . O . . . . . . . . . O . O . . .
. O . . O . . . . . . . . . O . . O . .
O O . . . . O O . . . O O . . . . O O .
. . . . . . O . . . . . O . . . . . . .
. . O O . . O . . . . . O . . O O . . .
. . O . . . O . . . . . O . . . O . . .
. . O . . . O . . . . . O . . . O . . .
. . O . . . O . . . . . O . . . O . . .
```

CPU time: 0.00702291 sec

score: 90

o. [Output] Program output is correct, good.

o. [Efficiency] can be improved.

life.c

```
1  // EE2310 lab11 The Game of Life
2  // 109061217, 林峻霆
3  // Date: 2020/12/18
4
5  #include "life.h"
6
7  void readGrid(CELL grid[N][N])
8      // A function that read data from input file and determine the initial
9      // state of each cell
10 {
11     int i, j;                                  // parameter for loop and index
12     char c;                                    // input char: . denote dead
13                                                //             o denote live
14     for (i = 0; i < N; i++) {
15         for (j = 0; j < N; j++) {
16             c = getchar();                     // read the valid input
17             while (c == ' ' || c == '\n') {
18                 c = getchar();
19             }
20             if (c == '.') {                    // check live or dead
21                 grid[i][j].next = DEAD;        // set the state
22                 grid[i][j].current = DEAD;
23                 grid[i][j].age = 0;
24                 grid[i][j].Nnbr = 0;
25             }
26             else {
27                 grid[i][j].next = LIVE;        // set the state
28                 grid[i][j].current = DEAD;
29                 grid[i][j].age = 0;
30                 grid[i][j].Nnbr = 0;
31                 grid[i][j].color =  GREEN;
32             }
33         }
34     }
35 }
36
37 int stillLife(CELL grid[N][N])
38     // A function that check whether the Still pattern occur
39     //      if occur, return 1
40     //      else, return 0
```

```
{
    int i, j;                       // parameter for loop and index
    int found = 1;                  // parameter for found still pattern

    for (i = 0; i < N; i++) {       // check whether current = next
        for (j = 0; j < N; j++) {
            if (grid[i][j].current != grid[i][j].next) {
                found = 0;
            }
            grid[i][j].current = grid[i][j].next;   // change to next state
        }
    }

    if (found)                      // Still pattern occur
        return 1;                   // return 1
    else                            // Still pattern not occur
        return 0;                   // return 0
}

void nextGen(CELL grid[N][N])
    // A function that determine the next state by calculating the amount of
    // alive neighbors.
    // It also change the age and color of each cell.
{
    int i, j;                               // parameter for loop and index
    int t, l;                               // parameter for neighbor calculate
    int row, col;                           // parameter for calculate row col

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            grid[i][j].Nnbr = 0;                // reset the amount
            for (t = -1; t <= 1; t++) {         // calculate the amount
                for (l = -1; l <= 1; l++) {
                    if (t != 0 || l != 0) {
                        row = (i + t + N) % N; // calculate row
                        col = (j + l + N) % N; // calculate col
                        if (grid[row][col].current == LIVE)
                            grid[i][j].Nnbr += 1;
                    }
                }
            }
```

```
82
83              if (grid[i][j].current == DEAD && grid[i][j].Nnbr == 3) {
84                  grid[i][j].next = LIVE;       // change from dead to alive
85              }
86              else if (grid[i][j].current == LIVE) {
87                  if (grid[i][j].Nnbr != 3 && grid[i][j].Nnbr != 2)
88                      grid[i][j].next = DEAD;   // change from alive to dead
89              }
90
91              if (grid[i][j].next == LIVE) {    // increase the age if alive
92                  grid[i][j].age += 1;
93              }
94              else {                            // reset age if dead
95                  grid[i][j].age = 0;
96              }
97
98              if (grid[i][j].age == 1) {        // color GREEN if 1-year-old
99                  grid[i][j].color = GREEN;
100             }
101             else if (grid[i][j].age == 2) {   // color YELLOW if 2-year-old
102                 grid[i][j].color = YELLOW;
103             }
104             else if (grid[i][j].age > 2) {    // color RED if > 2-year-old
105                 grid[i][j].color = RED;
106             }
107             else {                            // color white if dead
108                 grid[i][j].color = WHITE;
109             }
110         }
111     }
112 }
```