lab08

```
$ gcc lab08.c

$ ./a.out < sa.dat
Solution 1:
  3 6 1 | 2 9 4 | 7 8 5_
  9 2 8 | 3 7 5 | 1 4 6_
  5 7 4 | 1 6 8 | 2 3 9_
  ------|-------|------
  7 3 2 | 8 1 6 | 9 5 4_
  4 8 9 | 7 5 3 | 6 2 1_
  1 5 6 | 4 2 9 | 8 7 3_
  ------|-------|------
  6 9 3 | 5 8 2 | 4 1 7_
  8 1 5 | 6 4 7 | 3 9 2_
  2 4 7 | 9 3 1 | 5 6 8_
...
Solution 22:
  3 6 1 | 7 9 4 | 2 8 5_
  9 2 8 | 3 5 1 | 7 4 6_
  5 7 4 | 6 8 2 | 1 3 9_
  ------|-------|------
  7 3 2 | 8 1 6 | 9 5 4_
  4 8 9 | 5 7 3 | 6 2 1_
  1 5 6 | 4 2 9 | 8 7 3_
  ------|-------|------
  6 9 7 | 2 3 5 | 4 1 8_
  8 4 5 | 1 6 7 | 3 9 2_
  2 1 3 | 9 4 8 | 5 6 7_
Total number of solution found: 22.
Total number of solutions found: 22.
utime: 0.049629
```

---

score: 82.0

o. [Output] Program output is incorrect.

o. [Coding] lab08.c spelling errors: Sloutions(1), colunm(1), demended(1), determin(1), downmost(1), requirment(1)

o. [Format] Program format can be improved.

lab08.c

```c
// EE231002 Lob08. Finding Sudoku Sloutions
// 111060023, 黃柏霖
// 2022/11/14

#include <stdio.h>

#define N 9

int count = 0;                                  // count found solution

// to solve sudoku
void solve_sudoku(int A[N][N], int row, int col);
// to determin what can be in A[i][j]
int check(int A[N][N], int i, int j, int num);
// print sudoku
void print_sudoku(int A[N][N]);

int main(void)
{
    int i, j;                                   // loop control
    char tmp;                                   // store char temporary
    int M[N][N] = {0};                          // the sudoku

    for (i = 0; i < N; i++) {                   // read sodoku
        for (j = 0; j < N; j++) {
            scanf("%c ", &tmp);                 // store tmp
            // store char to int, and transfer '.' to 0
            M[i][j] = (tmp - '0' == '.' - '0') ? 0 : tmp - '0';
        }
    }
    solve_sudoku(M, 0, 0);                      // solve sudoku
    printf("Total number of solution found: %d.\n"
        , count);                               // print what's demended
            X,X count);                             // print what's demend
ed
    ',' should not lead a line
    return 0;
}

// to print sudoku
```

```
38 // int A[][]: the sudoku
39 // return nothing
40 void print_sudoku(int A[N][N])
41 {
42     int i, j;
43 
44     // print sudoku as the requirment
45     for (i = 0; i < N; i++) {
46         printf("  ");
47         if (i == 3 || i == 6) printf("------|-------|------\n  ");
48         for (j = 0; j < N; j++) {
49             if (j == 3 || j == 6) printf("| ");
50             printf("%d ", A[i][j]);
51         }
52         printf("\n");
53     }
54 }
55 
56 // to solve sudoku, count solution, and print it
57 // int A[N][N]: the sudoku
58 //     row, col: the row and colunm of sudoku
59 // return nothing
60 void solve_sudoku(int A[N][N], int row, int col)
61 {
62     int i;
63 
64     // find from up-left to down-right
65     // if not 0, don't fill in number
66     if (A[row][col] != 0) {
67         // found the right if it's still in sudoku
68         if (col < 8) solve_sudoku(A, row, col + 1);
69         // found the next row if it's still in sudoku
70         else if (row < 8) solve_sudoku(A, row + 1, 0);
71         // every position is searched
72         else {
73             count++;                                    // new solution found

  This line has more than 80 characters
74             printf("Solution %d:\n", count);
75             print_sudoku(A);                            // print sudoku
76         }
```

```
77      }
78      // if 0, then try which number can fill in
79      else {
80          for (i = 1; i <= N; i++) {
81              if (check(A, row, col, i)) {
82                  A[row][col] = i;                        // try through 1 to 9
83                  // not searching to the rightmost yet
84                  if (col < 8) {
85                      solve_sudoku(A, row, col + 1);      // found the right
86                      A[row][col] = 0;                    // trace back
87                  }
88                  // not searching to the downmost yet
89                  else if (row < 8) {
90                      solve_sudoku(A, row + 1, 0);        // found the next row
91                      A[row][col] = 0;                    // trace back
92                  }
93                  // every position is searched
94                  else {
95                      count++;                            // new solution found
96                      printf("Solution %d:\n", count);
97                      print_sudoku(A);                    // print sudoku
98                      A[row][col] = 0;                    // trace back
99                  }
100             }
101         }
102     }
103 }
104
105 // to check whether the num can be put in A[row][col]
106 // int A[N][N]: the sudoku
107 //     row, col: the tow and column of sudoku
108 //     num: the number that is checking
109 // return whether num can be put in
110 int check(int A[N][N], int row, int col, int num)
111 {
112     int k;                                      // loop control
113     int cell_x0 = row / 3 * 3;                  // the leftmost x of a cell
114     int cell_y0 = col / 3 * 3;                  // the uppermost y of a cell
115
116     for (k = 0; k < N; k++) {
117         if (A[row][k] == num) return 0;         // check row
```

```
118         if (A[k][col] == num) return 0;         // check column
119         if (A[cell_x0 + k % 3][cell_y0 + k / 3] == num)
120             return 0;                           // check cell
121     }
122     return 1;                                   // num is available
123 }
```