lab07

```
1  // EE2310 Lab07. Magic Square
2  // 108061213, 劉奕緯
3  // Oct. 31, 2019
   Need a blank line here
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <stdbool.h>
7  #define N 4                      // the size of square matrix
8  #define gross (N * N * N + N)/ 2// sum of each columns, rows, diagonals
   #define gross (N * N * N + N)/ 2 // sum of each columns, rows, diagonals
9
10 int M[N][N];                     // original matrix
11 int unused[N * N];               // list of unused number
12 int unusedqun = 0;               // # of unused number
13 int solution = 0;                // # of soltions
14 bool magiccheck();               // check if it is magic matrix
   bool magiccheck(void);              // check if it is magic matrix
15 void print_matrix();             // output the magic matrix
   void print_matrix(void);            // output the magic matrix
16 void read_matrix();              // intput the orginal matrix mainly
   void read_matrix(void);             // intput the orginal matrix mainly
17 void fill_entry();               // fill in unused numbers to matrix
   void fill_entry(void);              // fill in unused numbers to matrix
18 void permutate();                // permutate the unused list sequence
   void permutate(void);               // permutate the unused list sequence
   Need a blank line here
19 int main(void)
20 {
21     int matrix[N][N];
22
23     read_matrix();
24     for(;;) {
       for ( ; ; ) {
25         fill_entry(matrix);
26         if (magiccheck(matrix))
27             print_matrix(matrix);
28         permutate();             // permutate to try another
29     }
30     return 0;
31 }
   Need a blank line here
32 void fill_entry(int m[N][N])     // if origin entry is 0, fill in unused number
33 {                                // else be the same
34     int i, j, k = 0;
35
36     for (i = 0; i < N; i++)
37         for (j = 0; j < N; j++){
           for (j = 0; j < N; j++) {
```

```
38              if (M[i][j] == 0) m[i][j] = unused[k++];
39              else m[i][j] = M[i][j];
40          }
41  }
    Need a blank line here
42  void print_matrix (int m[N][N])
    void print_matrix(int m[N][N])
43  {
44      int i, j;
45
46      printf("Solution %d:\n", ++solution);
47      for (i = 0; i < N; i++) {
48          for (j = 0; j < N; j++)
49              printf("%4d", m[i][j]);
50          printf("\n");
51      }
52      printf("\n");
53  }
    Need a blank line here
54  void read_matrix()
    void read_matrix(void)
55  {
56      int i, j, k = 0;
57      int used[N * N];            // list of used numbers
58      bool exist;
59
60      for (i = 0; i < N; i++)     // input the origin matrix and list used numbers
61          for (j = 0; j < N; j++){
            for (j = 0; j < N; j++) {
62              scanf("%d" ,&M[i][j]);
                scanf("%d", &M[i][j]);
63              if (M[i][j]) used[k++] = M[i][j];
64          }
65      for (i = N * N; i > 0; i--){
        for (i = N * N; i > 0; i--) {
66                                  // list out the unused numbers
67          exist = false;
68          for (j = k; j>= 0; j--)
            for (j = k; j >= 0; j--)
69              if (used[j] == i) exist = true;
70          if (exist == false) unused[unusedqun++] = i;
71      }
72  }
    Need a blank line here
73  bool magiccheck(int m[N][N])
74  {
75      int i, j;
76      int sum = 0;                // sum of a row, column or diagonal
77
78      for (i = 0; i < N; i++)     // check diagonal's
```

```
79          sum += m[i][i];
80      if (sum != gross) return false;
81      sum = 0;
82      for (i = 0; i < N; i++)
83          sum += m[N - 1 - i][i];
84      if (sum != gross) return false;
85      sum = 0;
86      for (i = 0; i < N; i++){    // check rows
        for (i = 0; i < N; i++) {    // check rows
87          for (j = 0; j < N; j++)
88              sum += m[i][j];
89          if (sum != gross) return false;
90          sum = 0;
91      }
92      for (i = 0; i < N; i++){    // check columns
        for (i = 0; i < N; i++) {    // check columns
93          for (j = 0; j < N; j++)
94              sum += m[j][i];
95          if (sum != gross) return false;
96          sum = 0;
97      }
98      return true;
99  }
    Need a blank line here
100 void permutate(){                // Pandia permutation
    void permutate(void)                // Pandia permutation
    {
101     int i, j, k, l;
102     int temp;
103
104     for (j = unusedqun - 2; unused[j] <= unused[j + 1] && j >= 0; j--);
        for (j = unusedqun - 2; unused[j] <= unused[j + 1] && j >= 0; j--) ;
105                                 // find j
106     if (j < 0){                 // cannot permutate anymore. i.e. we had tried
        if (j < 0) {                 // cannot permutate anymore. i.e. we had tried
107                                 // all the matrixs. Therefore, we output the
108                                 // total number of solution and then end this
109                                 // program
110         printf("Number of solutions found: %d\n", solution);
111         exit(0);
112     }
113     for (k = unusedqun - 1; unused[k] >= unused[j] && k >= 0; k--);
        for (k = unusedqun - 1; unused[k] >= unused[j] && k >= 0; k--) ;
114                                 // find k
115                                 // swap
116     temp = unused[j];
117     unused[j] = unused[k];
118     unused[k] = temp;
119     for (i = j + 1, l = unusedqun - 1; i < l;  i++, l--){
        for (i = j + 1, l = unusedqun - 1; i < l;  i++, l--) {
```

```
120                                 // reverse
121         temp = unused[i];
122         unused[i] = unused[l];
123         unused[l] = temp;
124     }
125 }
```

[Format] can be improved.
[Coding] lab07.c spelling errors: intput(1), matrixs(1), orginal(1), soltions(1)

Score: 77