lab07

```
$ gcc -DN=11 lab07.c
lab07.c:65:1: warning: non-void function does not return a value [-Wreturn-type]
}
^
1 warning generated.

$ a.out
Latin Square 1:
  A B C D E
  B A D E C
  C D E B A
  D E A C B
  E C B A D
Latin Square 2:
  A B C E D
  B A D C E
  C D E A B
  D E A B C
  E C B D A
  ......
  ......
Latin Square 161280:
  E D C B A
  D E B A C
  C B A D E
  B A E C D
  A C D E B
Total number of Latin Squares found is 161280
CPU time: 0.707778 sec
```

---

score: 86.0

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

o. [Coding] lab07.c spelling errors: compling(1), storaged(1)

o. [Format] Program format can be improved.

o. [Compiler] warnings should be eliminated.

o. [Efficiency] can be improved.

lab07.c

```c
// EE231002 Lab07. Latin Squares
// 110060007, 黃俊穎
// 2021/11/22

#include <stdio.h>              // I/O library
#if !defined(N)                 // if N isn't defined, N = 3
#define N 3
#endif

char A[N][N];                   // Latin Square is an N * N matrix
int counter;                    // see the matrix is filled by alphabets
int num;                        // total found answer number

// a function creating a new matrix or fill symbols in rest elements
int create_fill(int row, int column, int n);

// a function judging if there are same symbols in that row or column
int decide(int row, int column);

// a function printing out results corresponding with Latin Squares
void print();
void print(void);

int main(void)                  // start main function
{
    create_fill(0, 0, N);       // call function to find all Latin Squares
    // print out total found answer
    printf("Total number of Latin Squares found is %d\n", num);
    return 0;
}

// we use counter to record alphabet number, continue filling in if current
// symbol is valid, or delete one and try for other alphabet
// until matrix is right, create new valid Latin squares
int create_fill(int row, int column, int n)
{
    int next_row;           // next row with respect to current row
    int next_column;        // next column with respect to current column
    char symb = 'A';        // initialize first symbol
    int i;                  // variable for loop
```

```
40
41      if (counter == n * n) {  // decide if matrix is filled up
42          print(n);            // print out the result
43          num++;               // counter of found answers
44      }else {
        } else {
45          for (i = 1; i <= n; i++) {
46              A[row][column] = symb;  // input symbol to matrix
47              symb++;          // change to next alphabet subsequently
48              counter++;       // if it reach to n * n, print out the
49                               // solution, or alphabet will keep be filled
50              // if repeated alphabet in same row and column,
51              // alphabets will be filled in same column first
52              if (decide(row, column)) {
53                  next_row = (row + 1) % n;
54                  next_column = column;
55                  // if that column is filled up, change column
56                  if (row == n - 1) {
57                      next_column = column + 1;
58                  }
59                  // if current array is right, create next matrix
60                  create_fill(next_row, next_column, n);
61              }
62              counter--;       // subtract invalid storaged number in matrix
63          }
64      }
65  }
66
67  // check if same row and column occur repeated alphabets respectively
68  int decide(int row, int column)
69  {
70      int i;                   // variable for loops
71      char decide_char;        // record current alphabet
72
73      decide_char = A[row][column];
74      // bound same row and check column
75      for (i = 0; i < column; i++) {
76          if (decide_char == A[row][i]) {
77              return 0;
78          }
79      }
```

```
80      // bound same column and check row
81      for (i = 0; i < row; i++) {
82          if (decide_char == A[i][column]) {
83              return 0;
84          }
85      }
86      return 1;                   // nothing error then return 1
87  }
88
89  // print out results compling with Latin Squares
90  void print(int n)
91  {
92      int i, j;
93
94      printf("Latin Square %d:\n", num + 1);
95
96      for (i = 0; i < n; i++) {
97          printf(" ");
98          for (j = 0; j < N; j++) {
99              printf(" %c", A[i][j]);
100         }
101         printf("\n");
102     }
103 }
```