lab06

```
$ gcc lab06.c
$ ./a.out
Solution 1:
 1 2 3 4 5
  1 2 3 4 5
 2 1 4 5 3
  2 1 4 5 3
 3 4 5 1 2
  3 4 5 1 2
 4 5 2 3 1
  4 5 2 3 1
 5 3 1 2 4
  5 3 1 2 4
...
...
Solution 161280:
 5 4 3 2 1
  5 4 3 2 1
 4 5 2 1 3
  4 5 2 1 3
 3 2 1 5 4
  3 2 1 5 4
 2 1 4 3 5
  2 1 4 3 5
 1 3 5 4 2
  1 3 5 4 2
Total number solutions found: 161280

CPU time: 0.611044 sec
score: 76
o. [Output] Program output is incorrect
o. [Format] Program format can be improved
o. [Coding] lab06.c spelling errors: jugde(1), neww(1), unqualifed(1)
o. [Efficiency] can still be improved.
```

lab06.c

```c
// EE231002 Lab06 Latin Squares
// 109061158. 簡佳吟
// Date: 2020/11/09

#include <stdio.h>
#define N 5

int A[N][N];            // array to test Latin Squares
long int Nsol = 0;      // number of Latin Squares found
int count = 0;          // number of elements filled in the square

void make(int x,int y);     // make each Latin Square
void make(int x, int y);     // make each Latin Square
int judge(int x, int y);    // jugde whether the Square are qualified
void print();               // print Square
void print(void);               // print Square
void init_array(int A[][N], int row, int col); // initialize array

int main(void) {
int main(void)
{
    init_array(A, 0,0);                             // initialize array
    init_array(A, 0, 0);                             // initialize array
    make(0, 0);                                     // make Square
    printf ("Total number solutions found: %ld\n", Nsol); //prompt
    printf("Total number solutions found: %ld\n", Nsol); // prompt
    return 0;                                       // done and return
}

void init_array(int A[][N], int row, int col) {
void init_array(int A[][N], int row, int col)
{
Comments?
    if (row < N && col < N) {
        A[row][col] = col + 1;              // fill numbers from 1 to 3 per row
        init_array(A, row , col + 1);
        init_array(A, row, col + 1);
    }
    if (col == N && row < N) {              // change to the next row
        init_array(A, row + 1, 0);          // and fill numbers
    }
}

void make(int x, int y) {
void make(int x, int y)
{
Comments?
    int i;                          // index
    int xx, yy;                     // index
Need a blank line here.
    if (count == N * N) {           // print the Latin Square
        print();                    // when filled the whole array
        Nsol++;
    }
    else {
        for (i = 1; i <= N; i++) {
            A[x][y] = i;            // assign neww number to array
            count++;
            if (judge(x, y)) {      // judge whether the new number
                                    // is different from other elements
                                    // int the same column or row
                yy =(y +1) % N;     // change to the next column in the
                yy = (y + 1) % N;     // change to the next column in the
                                    // same row
                if (y == N - 1) {   // when filled the last element in a row
                    xx = x + 1;     // change to the next row
                }
                else {
                    xx = x;         // otherwise, fill the same row
                }
                make(xx, yy);       // make another square
            }
            --count;                // if the number filled is unqualifed
                                    // let the count to the previous value
        }
   }
    }
}

int judge(int x, int y) {
int judge(int x, int y)
```

```
{
Comments?
    int i;                          // index
    int judgeN = A[x][y];           // assign judgeN
Need a blank line here.
    for (i = 0; i < y; i++) {
        if (judgeN == A[x][i]) {    // check the same row
            return 0;
        }
    }
    for (i = 0; i < x; i++) {       // check the same column
        if (judgeN == A[i][y]) {
           return 0;
        }
    }
    return 1;
}

void print() {
void print(void)
{
Comments?
    int i, j;                               // index
Need a blank line here.
    printf ("Solution %ld:\n", Nsol + 1);   // prompt
    printf("Solution %ld:\n", Nsol + 1);   // prompt
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            printf ("%2d", A[i][j]);        // print the Latin Square
            printf("%2d", A[i][j]);        // print the Latin Square
        }
        printf ("\n");
        printf("\n");
    }
}
```