lab07

```
$ gcc -DN=11 lab07.c
$ ./a.out < mat11.in
Matrix A is
   11 10 9 8 7 6 5 4 3 2 1
   10 11 10 9 8 7 6 5 4 3 2
   9 10 11 10 9 8 7 6 5 4 3
   8 9 10 11 10 9 8 7 6 5 4
   7 8 9 10 11 10 9 8 7 6 5
   6 7 8 9 10 11 10 9 8 7 6
   5 6 7 8 9 10 11 10 9 8 7
   4 5 6 7 8 9 10 11 10 9 8
   3 4 5 6 7 8 9 10 11 10 9
   2 3 4 5 6 7 8 9 10 11 10
   1 2 3 4 5 6 7 8 9 10 11
det(A) = 6144
CPU time: 1.40517 sec
score: 90
o. [Output] Program output is correct, good.
o. [Format] Program format can be improved
o. [Coding] lab07.c spelling errors: reult(1)
```

lab07.c

```
1  // EE2310 lab07 Matrix Determinants
2  // 109061217 林峻霆
   // 109061217, 林峻霆
3  // Date: 2020/11/16
   Need a blank line here.
4  #include <stdio.h>
5
6  #if !defined(N)
7  #define N 3
8  #endif
9
10 int Pandita(int P[N]);                          // declare Pandita function
11
12 int main(void)
13 {
14     int A[N][N];                                // a matrix for input
15     int i, j;                                   // parameter for loop
16     int det = 0;                                // the determinant
17     int factor = 1;                             // constant for multiply
18     int sgn = 1;                                // set initial sign value 1

   This line has more than 80 characters
19     int P[N];                                   // permutation array
20
21     for (i = 0; i < N; i++) {                   // setup permutation array
22         P[i] = i;
23     }
24     for (i = 0; i < N; i++) {                   // input the matrix
25         for (j = 0; j < N; j++) {
26             scanf("%d", &A[i][j]);
27         }
28     }
29
30     printf("Matrix A is\n");                    // print the matrix
31     for (i = 0; i < N; i++) {
32         printf("  ");
33         for (j = 0; j < N; j++) {
34             printf(" %d", A[i][j]);
35         }
36         printf("\n");
```

```
37      }
38
39      for (i = 0; i < N ; i++) {                  // calculate result of
        for (i = 0; i < N; i++) {                  // calculate result of
40          factor = factor * A[i][P[i]];           // first permutation
41      }
42      det += factor;                              // add result to determinant
43
44      sgn = sgn * Pandita(P);                     // sign of next permutation
45      while (sgn != 0) {                          // detect whether end loop
46          factor = 1;                             // reset factor
47          for (i = 0; i < N ; i++) {
            for (i = 0; i < N; i++) {
48              factor = factor * A[i][P[i]];       // calculate result of
49          }                                       // permutation
50          det += factor * sgn;                    // add reult to determinant
51          sgn = sgn * Pandita(P);                 // sign of next permutation
52      }
53
54      printf("det(A) = %d\n", det);               // print the result
55      return 0;                                   // end the program
56  }
57
58  int Pandita(int P[N])
59  // The Pandita function input an array. There is a variable "total" in it,
        // The Pandita function input an array. There is a variable "total" in it,
60  // which is used to record the amount of swap between this and last permutation
        // which is used to record the amount of swap between this and last permutat
ion
61  // Due to "The property of Odd and Even number", if total != 0 && total is even
        // Due to "The property of Odd and Even number", if total != 0 && total is e
ven
62  // then the sign of permutation will be equal to the last permutation.
        // then the sign of permutation will be equal to the last permutation.
63  //
        //
64  // To sum up, this function does the things below:
        // To sum up, this function does the things below:
65  // 1. return 0 if total = 0
        // 1. return 0 if total = 0
66  // 2. return 1 if total != 0 && total is even
```

```
    // 2. return 1 if total != 0 && total is even
// 3. return -1 if total != 0 && total is odd
    // 3. return -1 if total != 0 && total is odd
{
    int total = 0;                              // amount of swap
    int i ,j, k;                                // parameter for loop
    int i, j, k;                                // parameter for loop
    int tmp;                                    // parameter for swap
    int find = 1;                               // parameter for break

    i = N - 2;                                  // start from N-2 th index
    while (i >=0 && find) {
    while (i >= 0 && find) {
        if (P[i] < P[i + 1]) {                  // find corresponded index
            j = i + 1;
            total += 1;                         // renew the amount of swap
            for ( ; j < N; j++) {               // swap two index
                if (P[i] > P[j]) {
                    tmp = P[i];
                    P[i] = P[j - 1];
                    P[j - 1] = tmp;
                    j = N;
                }
                else if (j == N - 1) {
                    tmp = P[i];
                    P[i] = P[j];
                    P[j] = tmp;
                }
            }
            j = i + 1;                          // reverse from i+1 th index
            for (k = 0; j + k < N - 1 - k; k++) {
                total += 1;                     // renew amount of swap
                tmp = P[j + k];
                P[j + k] = P[N - 1 - k];
                P[N - 1 - k] = tmp;
            }
            find = 0;                           // jump out the loop
        }
        else                                    // if not found go to
            i = i - 1;                          // next index
    }
```

```
104     if (total == 0)                                 // check if total = 0
105         return 0;
106     else if (total % 2 == 0)                        // check if total is even
107         return 1;
108     else
109         return -1;
110 }
```