# lab08

```c
// EE2310 Lab08. Matrix Determinant
// 108081213, 劉奕緯
// Nov. 11, 2019

#include<stdio.h>
#include <stdio.h>
#if !defined(N)
#define N 3
#endif

double det(double A[N][N], int dim);    // caculate the determinant of a
                                        // "dim" * "dim" matrix "A"
void Minor(int i, int dim,  double minor[N][N], double M[N][N]);
                                        // get minor matrix "Minor"
                                        // by removing row 0 and column "i"
                                        // from a "dim" * "dim" matrix "M"
int main(void)
{
    int i, j;                           // loop index
    double Matrix[N][N];

    for (i = 0; i < N; i++)             // input matrix
        for (j = 0; j < N; j++)
            scanf("%lf", &Matrix[i][j]);
    printf("Matrix A is\n");            // output matrix
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++)
            printf("%3g", Matrix[i][j]);
        printf("\n");
    }
    printf("det(A) = %g\n", det(Matrix, N));
                                        // output answer
    return 0;
}

void Minor(int i, int dim,  double minor[N][N], double M[N][N])
                                          Need comments
{
    int m, n;                           // loop index
    int col;                            // index of column

    col = 0;
    for (n = 0; n < dim; n++) {
        if (n != i) {                   // Not copying i column
            for (m = 0; m < dim; m++)
                minor[m][col] = M[m + 1][n];
            col++;                      // next column
            }
```

```
            }
47      }
48  }
49
50
51  double det(double A[N][N], int dim)
                                                Need comments
52  {
53      double d = 0;                       // determinant
54      int sign = 1;                       // sign of cofactors
55      double minor[N][N];                 // to store minor matrix
56      int i;                              // loop index
57
58      if (dim == 1) return A[0][0];       // x * det(I(1*1) ) = a * 1
59      for (i = 0; i < dim; i++) {         // expansion using first row
60          Minor(i, dim, minor, A);        // get minor of A[0][i]
61                                          // and store in array"minor"
62          d += sign * A[0][i] * det(minor, dim - 1);
63          sign *= -1;                     // use the given formula
64      }
65      return d;
66  }
```

[CPU time] 1.25605 sec
[Format] can be improved.
[Coding] lab08.c spelling errors: caculate(1)
[Memory] efficiency can be improved.
[Recursion] should terminate when dim = 2

Score: 87