演算法 Homework 20191003
山下夏輝（Yamashita Natsuki）
R08922160

截止時間 星期四， 14:20　　**總分** 1　　**繳交** 線上輸入，網站網址，或者檔案上傳
接受繳交時間 10月3日 14:20 - 10月10日 14:20 7 天

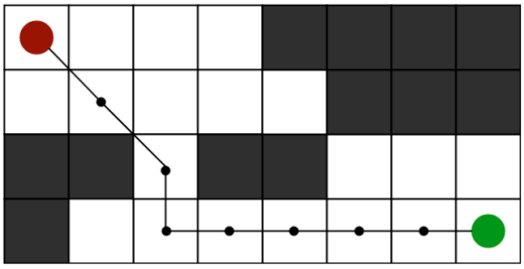
Given a road map, find the **number of ways** to reach the right bottom corner from the left top corner.

You can walk down, right, or lower-right in one step, but you can't walk into the obstacles.

You don't need to minimize the overall steps.

(1) (40%) Describe how you can solve the problem by *dynamic programming*.

(2) (60%) Write down the pseudo-code.



(1)

Variables m and n is the number of rows and column. A variable dp[i][j] is the number of ways to the position (i, j), i < m, j < n. The value of dp[m-1][n-1] is the number of ways to the right bottom corner.

The value of obstacle position is 0 even if the value of previous position is more than 1 because there is no way to reach obstacle.

The value of (1, 1) set 1 and the value of (i, j) is the sum of the values of (i-1, j), (i, j-1) and (i-1, j-1) which is the upper, left and left upper of (i, j). The example below shows that the value of (2,2) is 3. It means that there are 3 ways to reach (2,2). Also, there are 5 ways to reach (3,2). In this way, you can get the value of dp[m-1][n-1] which is the number of ways to the right bottom corner.

Example:

```
dp[i][j]:   [
            {1,  1,  1,  1,  0,  0,  0,  0}
            {1,  3,  5,  7,  8,  0,  0,  0}
            {0,  0,  8,  0,  0,  8,  8,  8}
            {0,  0,  8,  16, 16, 24, 40, 56}
            ]
```

(2)

```
NumberOfWays(m, n, matrix)
        // set the value of the left upper corner
        dp[0][0] = 1

        // init the first colomn
        for i = 1 to m
                if dp[i][0] != obstacle
                        dp[i][0] = dp[i-1][0]
                else
                        dp[i][0] = 0
        // init the first row
        for j = 1 to n
                if dp[0][j] != obstacle
                        dp[0][j] = dp[0][j-1]
                else
                        dp[0][j] = 0
        // sum the number of ways
        for i = 1 to m
                for j = 1 to n
                        if dp[i][j] != obstacle
                                dp[i][j] = dp[i-1][j] + dp[i][j-1] + dp[i-1][j-1]
                        else
                                dp[0][j] = 0

        return dp[m-1][n-1]
```