ADA Mini HW #3

Student Name: 林楷恩
Student ID: b07902075

1 Dynamic Programming Explanation

Let the map denoted by M, the rows numbered with 1 to N_{row} from top to bottom, and the columns numbered with 1 to N_{col} from left to right. Let `dp[i][j]` denotes the number of ways to reach position (i, j) from the left top corner (1, 1). Then I recursively define the values of dp[i][j]:

$$dp[i][j] = \begin{cases} 1 & if\ (i = 1\ and\ j = 1)\ or\ M[i][j]\ is\ a\ obstacle \\ dp[i-1][j] & if\ j = 1 \\ dp[i][j-1] & if\ i = 1 \\ dp[i-1][j] + dp[i][j-1] + dp[i-1][j-1] & if\ i > 1\ and\ j > 1 \end{cases}$$

Because we can go to (i, j) either from $(i-1, j)$ or $(i, j-1)$ or $(i-1, j-1)$ if it is in the map, and if there is obstacle on (i, j), (i, j) is unreachable so dp[i][j] = 0

2 Pseudo-code

```
input: a road map(2D array) M marked with some obstacles, N_row, N_col is the number of
       rows and columns respectively
output: the number of ways to reach the right bottom corner from the left top corner

function Solve(M):
    // initialize the boundary cases to zero for convenience.
    for i from 1 to N_row:
        dp[i][0] = 0
    for j from 1 to N_col:
        dp[0][j] = 0

    dp[0][0] = 1

    // fill the table by bottom-up method
    for i from 1 to N_row:
        for j from 1 to N_col:
            if is_obstacle ( M[i][j] ):
                dp[i][j] = 0
            else:
                dp[i][j] = dp[i-1][j] + dp[i][j-1] + dp[i-1][j-1]

    return dp[N_row][N_col]
```