COMP9101 Ass03
Name: Zanning Wang
zID: z5224151

2. You are given a 2D map consisting of an $C \times R$ grid of squares; in each square there is a number representing the elevation of the terrain at that square. Find a path going from square $(1, R)$ which is the top left corner of the map to square $(C, 1)$ in the lower right corner which from every square goes only to the square immediately below or to the square immediately to the right so that the number of moves from lower elevation to higher elevation along such a path is as small as possible. (20 pts)

Q2.

This problem similar to the Q1, we can also use dynamic programming to solve the question.
We use dp[i][j] to store when we get to the certain position, the lowest number of moves that we move from lower elevation to higher elevation.
We use cur[i][j] to represent current elevation of the terrain.

The base case dp[0][x], dp[y][0] ,which x∈[0, R], y∈[0, C]

There should have four situation in the recusion:
When current step higher the top one and the left one, we should increase 1 no matter go down or go right.
dp[i][j] => min(dp[i-1][j]+1, dp[i][j-1]+1) when cur[i][j] > cur[i-1] and cur[i][j] > cur[i][j-1]

When current step higher than the top one but smaller than the left one, we should increase 1 when go down
dp[i][j] => min(dp[i-1][j] + 1, dp[i][j-1]) when cur[i][j] > cur[i-1][j] and cur[i][j] <= cur[i][j-1]

When current step higher than the left one but smaller than the top one, we should increase 1 when go right.
dp[i][j] => min(dp[i-1][j], dp[i][j-1] + 1) when cur[i][j] <= cur[i-1] and cur[i][j] > cur[i][j-1]

When current step no higher than the top one and the left one, we should **NOT** increase 1 no matter go down or go right.
dp[i][j] => min(dp[i-1][j], dp[i][j-1]) when cur[i][j] <= cur[i-1] and cur[i][j] <= cur[i][j-1]

The recursion should show below:

$$dp[i][j] = \begin{cases} \min(dp[i-1][j]+1, dp[i][j-1]+1) & when\ \text{cur[i][j]} > \text{cur[i-1]} \text{ and cur[i][j]} > \text{cur[i][j-1]} \\ \min(\text{dp[i-1][j]} + 1, \text{dp[i][j-1]}) & \text{when cur[i][j]} > \text{cur[i-1][j]} \text{ and cur[i][j]} <= \text{cur[i][j-1]} \\ \min(\text{dp[i-1][j]}, \text{dp[i][j-1]}) & \text{when cur[i][j]} <= \text{cur[i-1]} \text{ and cur[i][j]} <= \text{cur[i][j-1]} \\ \min(\text{dp[i-1][j]}, \text{dp[i][j-1]}) & \text{when cur[i][j]} <= \text{cur[i-1]} \text{ and cur[i][j]} <= \text{cur[i][j-1]} \end{cases}$$

In order to find a path, we should store the cur[i][j] into a list, which is the final result, the time complexity would be O(C * R)