Q2

We allocate two-dimensional array DP_{r*c} to store all the possible outcomes. DP_{ij} means the number of all the unique path from $(0, 0)$ to (i, j). since we want to count the number of unique ways, we can assume that the length of each move is 1:

consider an arbitary allowed move in direction d and length of x. we can we can decompose it into x moves in direction d with length 1 and the route would be the same.

we first initialize $DP_{00} = 1$. it state(r,t) is an barrier , $DP_{rt} = 0$ is constant

consier an arbitary position $DP_{i,j}$:

1. if i=0 (DP_{0J}): it means we are in the first row, we can only reach this state if we move from $DP_{0,j-1}$ to $DP_{0,j}$.
2. if j=0 (DP_{i0}): it means we are in the first column, we can only reach this state if we move from $DP_{i-1,0}$ to $DP_{i,0}$.
3. o.w: there are three ways to reach the $state(i, j)$ so $DP_{ij} = DP_{i-1\,j} + DP_{i\,j-1} + DP_{i-1\,j-1}$

Implimentation:

In order to save space, we show the grid and DP in the same array. first we initialize all barrier as 1 and the rest of grid cells as 0. the we update the cells as below:

if $DP_{00} = 1$ the starting call is an barrier so there is no way to reach the final cell so we return 0.

first we set DP_{00}=1.

in the first row(i=0):

- if DP_{ij} is not an barrier: $DP_{ij_{new}} = DP_{i,j-1}$
- if DP_{ij} is an barrier: $DP_{ij_{new}} = 0$

we repeat the same algorithm for the first column(j=0)

then for the rest of the (r-1,c-1)grid:

- if DP_{ij} is not an barrier: $DP_{ij_{new}} = DP_{i-1\,j} + DP_{i\,j-1} + DP_{i-1\,j-1}$
- if DP_{ij} is an barrier: $DP_{ij_{new}} = 0$

we set the values of barrier to 0 becouse there is no way to reach them

```
def unique_paths(dp):
    r = len(dp)
    c = len(dp[0])
    if (dp[0][0]==1):
        return 0
    dp[0][0] = 1
    for j in range(1,c):
        if (dp[0][j] == 0):
            dp[0][j] = dp[0][j - 1]
        else:
            dp[0][j] = 0
    for i in range(1,r):
        if (dp[i][0] == 0):
            dp[i][0] = dp[i - 1][0]
        else:

            dp[i][0] = 0
    for i in range(1,r):
        for j in range(1,c):
            if (dp[i][j] == 0):
                dp[i][j] =dp[i - 1][j] + dp[i][j - 1] + dp[i-1][j - 1]
            else:
                dp[i][j] = 0

    return dp[r -1][c-1]
```

```
grid =  [[0 for x in range(6)] for x in range(6)]
grid[2][1]=1
grid[2][3]=1
grid[4][3]=1
print("the grid space:")
grid
```

```
the grid space:
[[0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0],
 [0, 1, 0, 1, 0, 0],
 [0, 0, 0, 0, 0, 0],
 [0, 0, 0, 1, 0, 0],
 [0, 0, 0, 0, 0, 0]]
```

```
sum_way=unique_paths(grid)
print("total ways to reach from A to B: ",sum_way)
```

```
total ways to reach from A to B:  318
```