Manhattan Ambulance
Problem ID: skeleton

When Albert Pierre Shannon (APS), a self-proclaimed utilitarian theorist, first came to New York, he was swooned by the rancid smell on the street and got carried over to a hospital by an ambulance. Lying on the stretcher in NYC traffic, he cannot help but dream the utilitarian dream that if the ambulances can knock over some pedestrians and stuff them right in, they could arrive at the hospitals faster and everyone will be taken care of. Being the smart person he is, APS starts to think about how practical his theory is. We know that each ambulance has a capacity k ($1 \leq k \leq 1000$), which means that the ambulance can carry up to k patients (not including APS).
Manhattan will be represented as a grid of $n * m$ cells ($1 \leq n \leq 100, 1 \leq m \leq 100$). Each cell has a number d ($1 \leq d \leq 10^9$), which is the distance from any of the four neighbouring cells to the cell. Each cell also has p ($0 < p \leq 1000$) pedestrians, which means that, if the ambulance choose to go to that cell, it would have to run over all p pedestrians and carry them all to the hospital. Roads are bidirectional and the ambulances can travel to in all four directions. The ambulance is initially empty and starts from the top left cell and wants to go to the bottom right cell.
Your job is to find the shortest path while making sure that the ambulances doesn't have to carry more than k patients on the ambulance en route.

Input

The first line has three integers n, m and k ($1 \leq n, m \leq 100, 1 \leq k \leq 1000$).
The next n lines will each contain m integers between 1 and 10^9 that describe the distance you have to travel in order to step onto the specific cell in the grid. The first number of the first line will always be 0.
The following n lines will each contain m integers between 1 and 1000 that describe the number of pedestrians on that cell. The first number on the first line will also always be 0.

Output

Output a single integer, the minimum distance you need to travel to get to the bottom-right cell without exceeding the capacity. If that's impossible, output -1.

Sample Input 1

```
2 2 3
0 2
2 1
0 1
1 2
```

Sample Output 1

```
3
```

Sample Input 2

```
3 3 4
0 1 2
2 1 2
2 1 1
0 1 1
1 1 1
1 1 1
```

Sample Output 2

```
4
```

Sample Input 3

```
5 5 8
0 1 1 1 1
2 2 2 2 1
1 1 1 1 1
1 2 2 2 2
1 1 1 1 1
0 2 2 2 2
1 1 1 1 2
2 2 2 2 2
2 1 1 1 1
2 2 2 2 2
```

Sample Output 3

```
-1
```