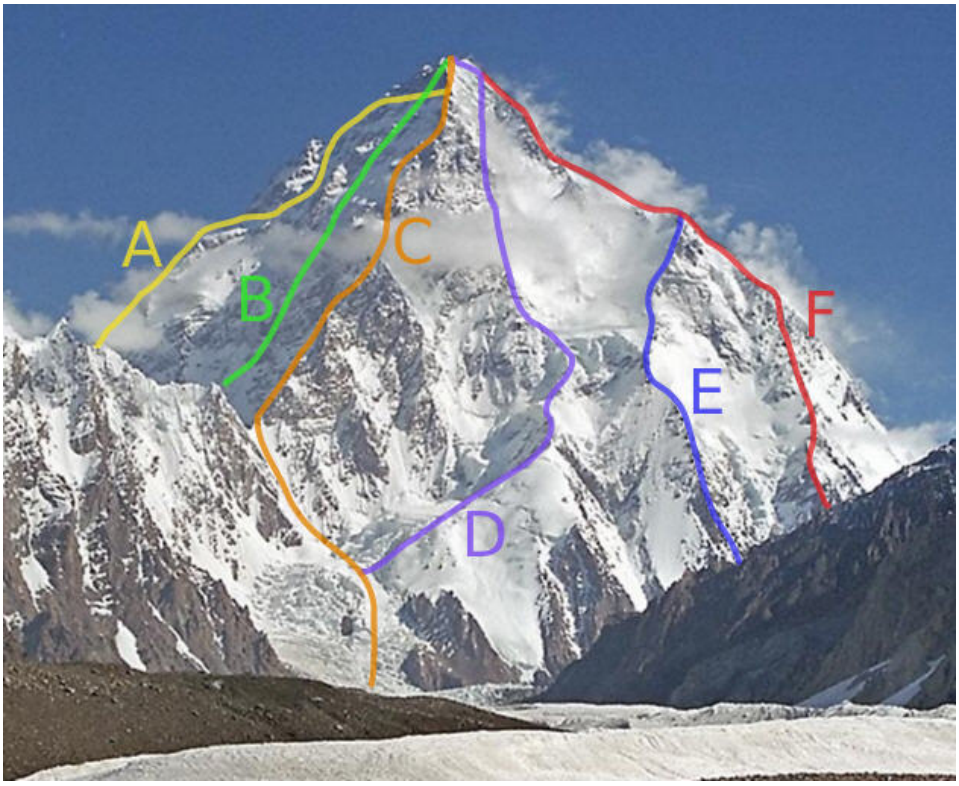
Problem_AM

K2 is the second-highest mountain on Earth. It is situated between Pakistan and China. It is known as the Savage Mountain due to the difficulty of ascent and has the second-highest fatality rate. It has never been climbed in winter. The major routes that have been climbed on the south side of the mountain are: A: West Ridge; B: West Face; C: Southwest Pillar; D: South Face; E: South-southeast Spur; and F: Abruzzi Spur, these are shown in the figure below:



A Pakistani mountaineer has estimated the risk associated with these routes. He has divided the entire scene of the south side of the mountain into small square blocks of equal size. Each square provides one handhold. The handhold has some risk associated with it. A new climber can use his estimate to determine the least dangerous route to the top. The climber should always start from the base of the mountain and from there move on to the block at the next level. As shown at Figure 1, at each step, the climber has the option to

continue climbing the mountain by moving on to one out of three possible blocks, meaning the block located diagonally on right of the current block, the block located diagonally on the left or the block located exactly above the current block. Note that in cases where the climber is at the corner of a column, some handholds may not be available, for example diagonal-right handhold when the climber is already at the handhold of the right-most block.

Fig. 1. Graphical illustration of the possible moves of the climber. The current block of the climber is highlighted in orange whereas all the possible next moves are depicted with blue arrows.

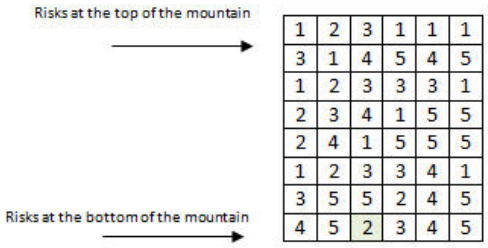
Task

In this problem a two dimensional table describing the risks at each position of the climbing phase would be given as input to the program and the least dangerous possible route for reaching the top of the mountain will have to be estimated. The risks in the table are defined in a bottom-up order meaning that the first row of the table will signify the risks associated with the individual positions at the top of the mountain whereas the values at the last row of the table show the corresponding risk for all the available blocks at the base of the mountain.

For any given route (i.e. path starting from the base of the mountain and leading to its top) the overall risk associated with it will be calculated as the sum of danger ratings (costs) of all the individual blocks used for that path.

```
1 2 3 1 1 1
3 1 4 5 4 5
1 2 3 3 3 1
2 3 4 1 5 5
2 4 1 5 5 5
1 2 3 3 4 1
3 5 5 2 4 5
4 5 2 3 4 5
```

For a better understanding, please consider the following exemplary two dimensional array being given as input to the program.



1	2	3	1	1	1
3	1	4	5	4	5
1	2	3	3	3	1
2	3	4	1	5	5
2	4	1	5	5	5
1	2	3	3	4	1
3	5	5	2	4	5
4	5	2	3	4	5

Fig. 2 An example of all the risks associated with every single block that may be followed to reach the top of the mountain.

Assuming a zero-based indexing scheme, the program should start from the base of the mountain (the last row), choosing a block at each level where the possible blocks to climb from a particular position have already been described above (see figure 1). The goal of the program is to find an “optimal” path to the top of the mountain (the first row) i.e. one that has least risk attached to it.

NOTE:
If, at any cell, there are multiple candidates for the handholds at the next level (each of which leads to a path with the same overall risk factor), then the order of preference for handholds is: diagonal-left > exactly-above > diagonal-right. For example, if at a particular cell, selecting either the handhold at the next level which is exactly-above or diagonally to the right of the current one leads to the “optimal” path (but not the one which is diagonally to the left), you should proceed by moving onto the cell that is exactly above the current one, since it’s preference is higher than the one to the diagonal-right. In this way, your program should always arrive at a unique minimal risk path.

Input

The program shall receive the following input from the standard input stream:

- The first line of the input will be two positive integer numbers [M, N] separated by a space
- Then M lines should follow, each one containing N positive space separated integer values representing the risk associated in each block of the mountain from top to the base of the row.

An example of the input format is provided below:

```
12 10
3 1 1 3 1 4 0 3 3 2
0 0 1 5 0 2 3 2 4 3
1 3 2 1 4 1 1 2 1 4
1 1 4 2 0 1 5 5 1 1
0 4 0 1 1 4 3 1 1 5
4 1 0 5 2 4 1 4 0 4
0 3 3 1 3 1 1 0 2 2
1 3 1 0 2 3 1 0 1 5
3 3 3 5 1 3 3 1 5 0
4 2 0 4 0 4 2 5 0 4
4 1 2 2 3 2 2 5 4 1
4 1 2 4 1 3 1 1 1 2
```

Output

The program should output to the standard output stream the complete path with the least associated risk that a mountaineer should follow to reach the top of the mountain. Then a new line should follow describing the total accumulated risk of the reported path. The output should be formatted as follows:

```
Minimum risk path = [R1,C1][R2,C2]…[RK,CK]
Risks along the path = R
```

where R1, C1, R2, C2… Rk, Ck are the **zero-based** indices of the cells of the two dimensional array that were followed to reach the mountain's peak using this route. The first pair [R1,C1] indicates the cell at the top of the mountain whereas the last pair [Rk,Ck] denotes the base. The first index in the pair always stands for the row and the second for the column of each individual cell, and they are separated **ONLY** by a comma. Also, each pair of indices is enclosed in square brackets and there is **NO** space between adjacent pairs. The value R is the integer value corresponding to the overall risk associated to the reported route, calculated as the sum of all the costs of the blocks used for that path.

NOTE:
The output should consist of only 2 lines. There is no new third line in the output.

An example of the output is provided below:

```
Minimum risk path = [0,1][1,0][2,0][3,1][4,2][5,2][6,3][7,3][8,4][9,4][10,3][
11,4]
Risks along the path = 8
```

Sample Input:

```
7 7
2 2 2 1 2 3 4
2 2 1 2 1 2 5
4 1 1 2 1 1 5
1 2 2 3 3 4 1
4 1 1 2 2 1 4
3 4 1 4 1 2 1
1 2 1 1 1 2 1
```

Sample Output:

```
Minimum risk path = [0,3][1,2][2,1][3,0][4,1][5,2][6,2]
```

```
Risks along the path = 7
```

Problem Author: IEEE

Suggest Edits

Emacs | Normal | Vim

Select Language: C++

save code

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;


int main() {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    return 0;
}

```

Line: 1 Col: 1 Count: 227

☐ **Use a custom test case**

Upload Code as File

Compile & Test | Submit Code