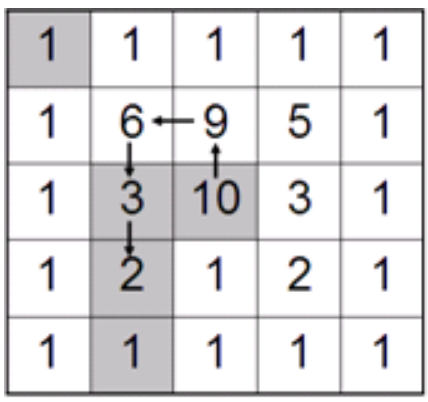
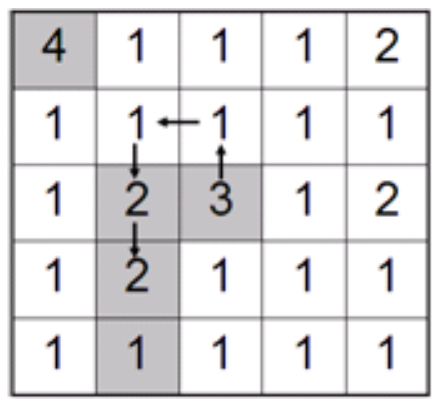
The block world is divided into n rows n columns of blocks. Rows are numbered 1 to n from top to bottom, and columns are numbered 1 to n from left to right. You start from a square, follow a strictly height-decreasing path until no legal moves are possible. During your trip, you can use a jewel magnetizer to get jewels not too far away. To be precise, you can get a jewel in square (r_1, c_1) if and only if there is a square (r_2, c_2) on your path such that $\max\{|r_1 - r_2|, |c_1 - c_2|\} \leq r$.

There is one thing you should be aware of: your bag capacity is limited, so you can only get at most m jewels. There is at most one jewel on each block.



(a) height



(b) jewel value

The picture above shows an example. The numbers in the left picture describes the heights in each block, where the numbers in the right picture describes the jewel values in each block. An optimal solution for $m = 5$ and $r = 1$ is shown in the figures. The arrows denote the path, gray squares corresponds to jewels you should get. The total value is 4+3+2+2+1=12.

Write a program to get the highest total value of jewels.

Input

The input consists of at most 30 test cases. Each case begins with a line containing three non-negative integers n, m and r ($1 < n < 21, 0 < m < 101, r < 6$), where n is the size of the block world, m is the bag capacity, r is the range of jewel magnetizer. The second line contains two integers $r0$, $c0$ ($1 \leq r0, c0 \leq n$), the row number and column number of the initial position. The next n lines each contain n non-negative integers not greater than 8000, the heights of each block. The next n lines each contain n non-negative integers not greater than 1000, the value of jewel at each block (zero means there is no jewel). The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and the highest value.

Sample Input

```
5 5 1
3 3
1 1 1 1 1
1 6 9 5 1
1 3 10 3 1
1 2 1 2 1
1 1 1 1 1
4 1 1 1 2
1 1 1 1 1
1 2 3 1 2
1 2 1 1 1
1 1 1 1 1
4 2 1
1 3
4 5 6 9
3 9 9 9
2 1 1 9
9 9 9 9
0 0 0 1
0 0 0 0
0 0 0 0
0 0 1 5
0
```

Sample Output

```
Case 1: 12
Case 2: 2
```