



## [I] Kind of a Blur

Program:	blur.(c cpp java)
Input:	blur.in
Balloon Color:	Silver

### Description

Image blurring occurs when the object being captured is out of the camera's focus. The top two figures on the right are an example of an image and its blurred version. Restoring the original image given only the blurred version is one of the most interesting topics in image processing. This process is called deblurring, which will be your task for this problem.

In this problem, all images are in grey-scale (no colours). Images are represented as a 2 dimensional matrix of real numbers, where each cell corresponds to the brightness of the corresponding pixel. Although not mathematically accurate, one way to describe a blurred image is through averaging *all the pixels that are within (less than or equal to) a certain Manhattan distance<sup>†</sup> from each pixel (including the pixel itself)*. Here's an example of how to calculate the blurring of a 3x3 image with a blurring distance of 1:

$$\begin{aligned} & \text{blur} \left( \begin{bmatrix} 2 & 30 & 17 \\ 25 & 7 & 13 \\ 14 & 0 & 35 \end{bmatrix} \right) \\ &= \begin{bmatrix} \frac{2+30+25}{3} & \frac{2+30+17+7}{4} & \frac{30+17+13}{3} \\ \frac{2+25+7+14}{4} & \frac{30+25+7+13+0}{5} & \frac{17+7+13+35}{4} \\ \frac{25+14+0}{3} & \frac{7+14+0+35}{4} & \frac{13+0+35}{3} \end{bmatrix} \\ &= \begin{bmatrix} 19 & 14 & 20 \\ 12 & 15 & 18 \\ 13 & 14 & 16 \end{bmatrix} \end{aligned}$$



Focused Image



Blurred Image

6	5	4	3	4	5	6
5	4	3	2	3	4	5
4	3	2	1	2	3	4
3	2	1	0	1	2	3
4	3	2	1	2	3	4
5	4	3	2	3	4	5
6	5	4	3	4	5	6

Manhattan Distance

Given the blurred version of an image, we are interested in reconstructing the original version assuming that the image was blurred as explained above.

<sup>†</sup>The Manhattan Distance (sometimes called the Taxicab distance) between two points is the sum of the (absolute) difference of their coordinates. The grid on the lower right illustrates the Manhattan distances from the grayed cell.

### Input Format

Input consists of several test cases. Each case is specified on  $H + 1$  lines. The first line specifies three non negative integers specifying the width  $W$ , the height  $H$  of the blurred image and the blurring distance  $D$  respectively where  $(1 \leq W, H \leq 10)$  and  $(D \leq \min(W/2, H/2))$ . The remaining  $H$  lines specify the gray-level of each pixel in the blurred image. Each line specifies  $W$  non-negative real numbers given up to the 2<sup>nd</sup> decimal place. The value of all the given real numbers will be less than 100.

Zero or more lines (made entirely of white spaces) may appear between cases. The last line of the input file consists of three zeros.

### Output Format

For each test case, print a  $W \times H$  matrix of real numbers specifying the deblurred version of the image. Each element in the matrix should be approximated to 2 decimal places and right justified in a field of width 8. Separate the output of each two consecutive test cases by an empty line. Do not print an empty line after the last test case. It is guaranteed that there is exactly one unique solution for every test case.

### Sample Input/Output

```
blur.in
2 2 1
1 1
1 1

3 3 1
19 14 20
12 15 18
13 14 16

4 4 2
14 15 14 15
14 15 14 15
14 15 14 15
14 15 14 15

0 0 0
```

```
OUTPUT
1.00 1.00
1.00 1.00

2.00 30.00 17.00
25.00 7.00 13.00
14.00 0.00 35.00

1.00 27.00 2.00 28.00
21.00 12.00 17.00 8.00
21.00 12.00 17.00 8.00
1.00 27.00 2.00 28.00
```