Problems

A. High-Pass Filter
B. Image Scaling
C. Run-length Encoding
D. Maximal Area
E. LATIN VOWEL CHARACTERS
F. Magic Mirror
G. Greedy Eat
H. Highly Abundant Number
And more….

Advice, hints, and general information

- The problems are not sorted by difficulty.
- Your solution programs must read input from *standard input* (e.g. System.in in Java or cin in C++) and write output to *standard output* (e.g. System.out in Java or cout in C++). For further details and examples, please refer to your administrator guide and Domjudge documentation.
- For information about which compiler flags and versions are used, please refer to your administrator guide. (Python 2.7.17, Oracle Java 1.8.0_144, gcc 7.5.0 (C, C++ std14)).
- Your submissions will be run multiple times, on several different inputs. If your submission is incorrect, the error message you get will be the error exhibited on the first input on which you failed.
    - E.g., if your instance is prone to crash but also incorrect, your submission may be judged as either "Wrong Answer" or "Run Time Error", depending on which is discovered first. The inputs for a problem will always be tested in the same order.
- If you think some problem is ambiguous or underspecified, you may ask the judges for a clarification request through the Domjudge system. The most likely response is "No comment, read problem statement", indicating that the answer can be deduced by carefully reading the problem statement or by checking the sample test cases given in the problem, or that the answer to the question is simply irrelevant to solving the problem.
- In general, we are lenient with small formatting errors in the output, in particular whitespace errors within reason, and upper/lower case errors are often (but not always) ignored. But not printing any spaces at all (e.g. missing the space in the string "1 2" so that it becomes "12") is typically not accepted. The safest way to get accepted is to follow the output format exactly.
- For problems with floating point output, we only require that your output is correct up to some error tolerance. For example, if the problem requires the output to be within either absolute or relative error of 10^{-4}, this means that
    - If the correct answer is 0.05, any answer between 0.0499 and .0501 will be accepted.
    - If the correct answer is 500, any answer between 499.95 and 500.05 will be accepted.
- Any reasonable format for floating point numbers is acceptable. For instance, "17.000000", "0.17e2", and "17" are all acceptable ways of formatting the number 17. For the definition of reasonable, please use your common sense.

Problem A
High-Pass Filter

Time Limit: 3 seconds
Memory Limit: 512 Megabytes

Problem description

High-pass filter (HPF) is an electronic filter that passes signals with frequency higher than a certain cutoff frequency and attenuates signals with frequencies lower than the cutoff frequency. The amount of attenuation for each frequency depends on the filter design. A high-pass filter is usually modeled as a linear time-invariant system. It is sometimes called a low-cut filter or bass-cut filter in the context of audio engineering.

In image processing, image filtering is useful for many applications, including smoothing, sharpening, removing noise, and edge detection. A filter is defined by a kernel, which is a small array applied to each pixel and its neighbors within an image. In most applications, the center of the kernel is aligned with the current pixel, and is a square with an odd number (3, 5, 7, etc.) of elements in each dimension. The process used to apply filters to an image is known as *convolution*, and may be applied in either the spatial or frequency domain.

Nam has many grayscale Images which are represented as a matrix with m rows and n columns where m, n is the number of pixels in each sides of the images. To sharpen these images, Nam decides to use a high-pass filter in his own image processing tool. Help him to implement high-pass filter function with following kernel:

$$\begin{bmatrix} -1/9 & -1/9 & -1/9 \\ -1/9 & 8/9 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{bmatrix}$$

Be noticed that for simplifying the problem, we will apply convolution with zero padding edges approach.

Input

The first line contains two integers: m, n ($1 < m, n \leqslant 1024$), the number rows and columns of the original image matrix.

The next m lines contain n integer numbers a_{ij} where $0 \leqslant a_{ij} \leqslant 255$ - the grey value of the pixel in the original image separated by spaces ($0 \leqslant i \leqslant m, 0 \leqslant j \leqslant n$).

Output

The m lines contain n integer numbers b_{ij} where $0 \leqslant b_{ij} \leqslant 255$ - the grey value of the pixel in the filtered image separated by spaces ($0 \leqslant i \leqslant m, 0 \leqslant j \leqslant n$).

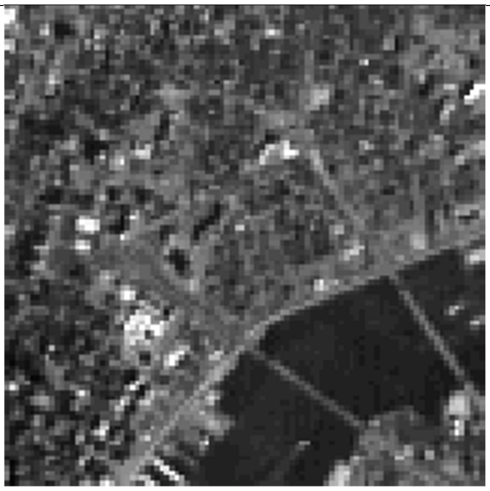
Figure A. 1.Original image

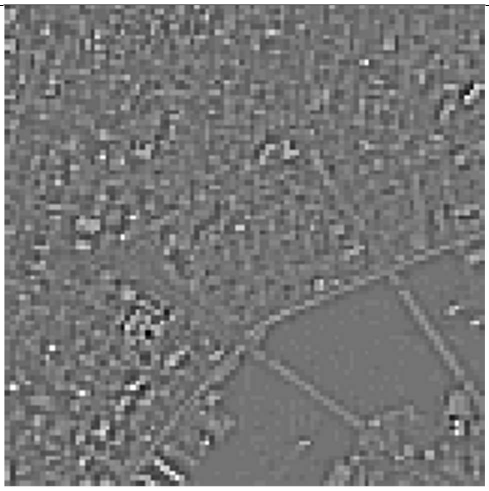
Figure A. 2.High-pass filtered image

Example:

Input	Output

3 3
9 9 9
9 9 9
9 9 9
	5 3 5
	3 0 3
	5 3 5

Input	Output

4 4
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
	0 0 0 0
	0 0 0 0
	0 0 0 0
	0 0 0 0

Input	Output

3 3
9 18 9
18 27 18
9 18 9
	1 7 1
	7 12 7
	1 7 1