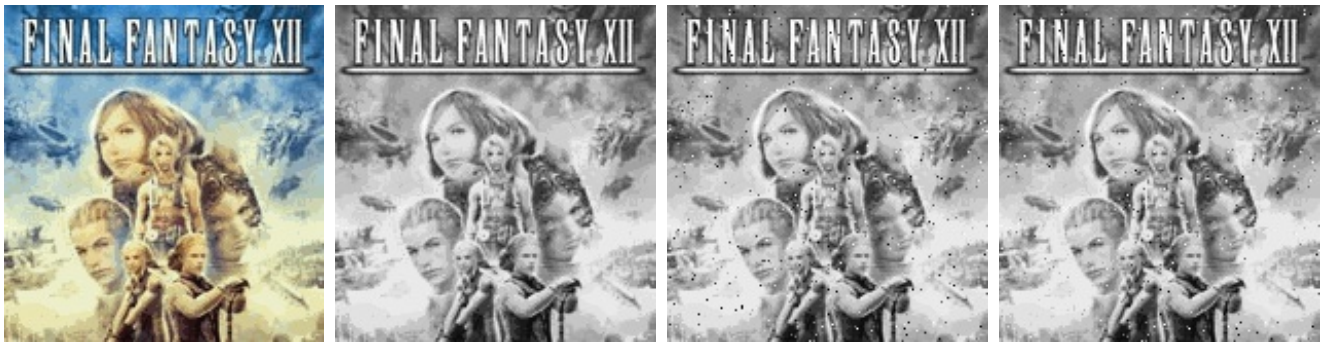
Digital Image Processing

One of the most interesting problems of contemporary times is digital image processing to remove noise. A good solution to this problem is very important e.g. when developing digital cameras. In this task we are given a set of pictures, each of which is a grayscale image, transferred by some communication channel with failures. During the transfer some data was corrupted. A picture is defined as a rectangular matrix of integers from the range from 0 (black) to 255 (white). A number X at position (i, j) means that the pixel in the picture at the point with coordinates x = i and y = j has color RGB(X, X, X). The considered form of corruption generates noise in the following way: each pixel of the picture has its color replaced with probability between 2 and 20% by a random value from the range [0; 255].

Thus, you now receive a set of corrupted pictures, which were originally e.g. avatars, banners or photos. You are to restore the picture with maximum quality. The more exact a picture you obtain, the fewer penalty points you get.

Original Picture | Grayscale Picture | Noised Picture | Denoised Picture



Input

t – the number of test cases [*t* <= 60] (total number about 250), then *t* test cases follows.
Each test case begins with three integers: *Q*, *H* and *W*, denoting the noise probability for the generator in percent, and the height and width of picture respectively [2 <= *Q* <= 20], [10 <= *H, W* <= 200]. Then *H* rows follow with *W* integers in each of them.

Output

For each test case you must output a picture after noise reduction in following format: In the first line output the two integers *H* and *W*. Then *H* rows must follow with *W* integers in each of them. Each integer is the color value of a pixel after the restoration process.

Score

The score of the program will be equal to the sum of scores for each single test case + 1. The number of points for a single test case is calculated from the formula:
score = sqrt((x[1][1]1 - x[1][1]2)^2 + (x[1][2]1 - x[1][2]2)^2 + ... + (x[H][W]1-x[H][W]2)^2),
where x[i][j]1 is the color of the pixel at position i, j in the restored picture

and x[i][j]2 is the color of the pixel at position i, j in the original picture.

Example

Input:

1
6 20 20
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 028 255 255
255 255 200 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 000 255 255 096 255 079 079 079 079 255 045 255 045 255 068 043 043 043 255
255 000 255 255 096 255 079 255 255 255 255 045 129 045 255 068 255 255 043 255
255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 189 255
255 058 058 058 096 255 079 079 079 079 255 045 255 045 255 068 255 255 068 255
255 076 255 255 096 255 079 255 255 255 255 185 255 045 255 068 255 255 068 255
255 000 255 255 096 255 079 255 242 255 255 045 255 045 255 068 255 255 068 255
255 000 255 255 096 255 079 079 079 079 255 045 255 043 255 048 048 048 048 255
255 255 255 255 255 255 255 255 058 255 255 255 198 255 255 255 255 255 255 255
036 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 100 195 002 167 002 002 002 002 002 002 002 002 002 002 255 255 255 255
255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 088 255 255 255 255
255 255 255 255 002 002 002 002 002 002 046 002 002 002 002 002 255 255 143 255
255 255 255 255 002 002 002 002 002 013 002 002 002 002 002 002 255 255 255 255
255 255 177 255 255 255 255 104 255 255 255 255 255 255 255 255 012 133 255 255
022 022 022 022 066 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 079 079 079 079 079 079 079 079 079 079 079 079 079 079 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 203 255 255

Output:

20 20
253 253 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
254 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 000 255 255 096 255 079 079 079 079 255 045 255 045 255 068 043 043 043 255
255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 043 255
255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 043 255
255 058 058 058 096 255 079 079 079 079 255 045 255 045 255 068 255 255 068 255
255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 068 255
255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 068 255
255 000 255 255 096 255 079 079 079 079 255 045 255 043 255 048 048 048 048 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 079 079 079 079 079 079 079 079 079 079 079 079 079 079 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255

Score:

Original picture:
20 20
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 000 255 255 096 255 079 079 079 079 255 045 255 045 255 068 043 043 043 255
255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 043 255
255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 043 255

255 058 058 058 096 255 079 079 079 079 255 045 255 045 255 068 255 255 068 255
255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 068 255
255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 068 255
255 000 255 255 096 255 079 079 079 079 255 045 255 043 255 048 048 048 048 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 079 079 079 079 079 079 079 079 079 079 079 079 079 079 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255

score = sqrt(2^2 + 2^2 + 1^2) + 1 = 1 + 3 = 4 (three pixels differ in the top-left corner)

Original picture: 

Noisy picture: