Image Upsampling

To reduce the size of a 2-dimensional image, I, containing R rows and C columns of pixels, the image is downsampled using a crude algorithm.

The downsampling algorithm begins sampling from the top-left pixel position, $(0, 0)$, of the original image and then proceeds to retain only those pixels which are located in those positions where both the row number and the column number are either 0, or integer multiples of some integer N. The downsampled image only contains r rows and c columns where these values correspond to $1 + floor((R - 1)/N)$ and $1 + floor((C - 1)/N)$

Let's assume for a moment that the original image is as follows (where each character indicates a pixel):

```
ABCDEFG
HIJKLMN
OPQRSTU
VWXYZab
cdefghi
jklmnop
```

This image has 6 rows and 7 columns.
Assume that the pixel $\mathbf{A}$ is the one positioned at $row = 0, column = 0$.
If we downsample this image using $N = 2$, we only retain those pixels located in positions where both the row and column numbers are even. This means the downsampled image will be:

```
ACEG
OQSU
cegi
```

Observe that when $N = 2$, the downsampled image retains over a quarter of the pixels.

Now, if we downsample this image using $N = 3$, we only retain those pixels located in positions where both the row and column numbers are multiples of 3. This means the downsampled image will be:

```
ADG
VYb
```

Observe that when $N = 3$, the downsampled image retains only 6 pixels.

Task

You are provided with the RGB values for a downsampled image and the downsampling coefficient (N). Given the size of the original image (and assuming that the algorithm used for downsampling is the one described above), restore the original image using the interpolation or upsampling algorithm of your choice.

Input Format

The first line contains 3 space-separated integers, r (the number of rows in the downsampled image), c (the number of columns in the downsampled image) and N (the downsample coefficient), respectively.

The second line contains 2 space-separated integers, R and C, representing the respective numbers of rows and columns in the original image.

The R subsequent lines describe the $2D$ grid representing the pixel-wise values from the images (which were originally in *JPG* or *PNG* formats).

Each line contains C pixels, and each pixel is represented by three comma-separated values in the range from 0 to 255 denoting the respective *Blue*, *Green*, and *Red* components. There is a space between successive pixels in the same row.

No input test case will exceed 3MB in size. This is the size of the RGB test matrix, NOT the original image from which it was generated.

Scoring

We define the $distance$ between an expected pixel value (r, g, b) and an estimated pixel value (r', g', b') to be: $\sqrt{\frac{(r-r')^2+(g-g')^2+(b-b')^2}{3}}$

For each test image, we compute M (the mean distance between the actual pixel value and the estimated value) for each pixel position in the image.

The score awarded for an image is $max(0, \frac{100-M}{100})$. This means that you will receive no score for a test if your mean $distance$ from the actual image exceeds 100. Your final score is the average of the awarded scores for each test image.

You may make no more than 15 submissions for this problem, during the contest.

Output Format

Print a $2D$ grid of pixel values describing the upsampled image. Your output should follow the same format as the grid received as input.

Note: You should *only* print the grid; there is no need to specify the number of rows and columns here.

Sample Input

```
3 3 2
6 6
0,0,200 0,0,10 10,0,0
90,90,50 90,90,10 255,255,255
100,100,88 80,80,80 15,75,255
```

The downsampled image is 3×3 pixels, with a sampling coefficient of 2. You must upsample this image and generate an interpolated image of 6×6 pixels.

The image described above is represented by 3×3 pixels. The *Blue*, *Green*, and *Red* values provided for each pixel are comma-separated, and the pixel descriptions are space-separated.

The top-left pixel is defined as $Blue = 0, Green = 0, Red = 200$.
The top-right pixel is defined as $Blue = 10, Green = 0, Red = 0$.
The bottom-right pixel is defined as $Blue = 15, Green = 75, Red = 255$.
The bottom-left pixel is defined as $Blue = 100, Green = 100, Red = 88$.

Sample Output

```
0,0,200 100,100,100 0,0,10 5,5,5 10,0,0 50,50,50
90,90,50 90,90,50 90,90,10 90,90,10 255,255,255 0,0,10
100,100,88 100,100,88 80,80,80 80,80,80 15,75,255 15,75,255
100,100,88 100,100,88 80,80,80 80,80,80 15,75,255 15,75,255
100,100,88 100,100,88 80,80,80 80,80,80 15,75,255 15,75,255
90,90,50 90,90,50 90,90,10 90,90,10 255,255,255 0,0,10
```

This is for the purpose of explanation only. We have generated an image of size $6x \times 6$ from the input image.