

You will be given a three channel image (3 colors per pixel: red, green and blue with values between 0 and 255) that you have to compress and then reconstruct as accurately as possible. However, there are two catches:

1. The compression is only in terms of the colors. In the compressed form each pixel must have one of 16 colors chosen by you
2. Before you have a chance to reconstruct the image, your compression will be altered. A **rectangular section of it will be cut** and given to you. Some of the pixels in this area will **swap their color with one of their neighbours**.

Your solution will be executed twice:

1. On the first run you will get the original image in `imrec.in` and will have to compress it and put it in `imrec.out`
2. On the second run you will get the altered compressed image in `imrec.in` and will have to try and reconstruct the original image only in **the rectangular area corresponding to the cut made on your compression**.

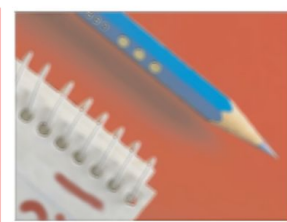
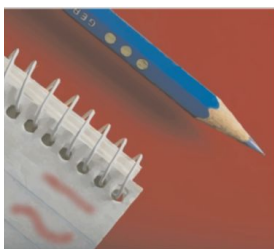
See the input/output section for more details.

Original image

Example compressed image (created by your solution)

Compressed image after the alteration

Reconstructed image (created by your solution)



Scoring

The score of your solution will be the difference of the reconstructed image and **its corresponding area** in the original image. The difference between two images is defined as the sum of the differences of every two corresponding pixels. The difference of two pixels is defined by the formula:

$$(r_1-r_2)^2 + (g_1-g_2)^2 + (b_1-b_2)^2$$

where r_1 and r_2 are the red values of the two pixels, g_1 and g_2 are the green values and b_1 and b_2 for the blue values.

You will receive $(minScore + 1) \div (yourScore + 1) * 100\%$ percent of the max points for the current test. *yourScore* is (unsurprisingly) your score and *minScore* is the lowest score that any solution got for that test.

Input / Output

For input use the file `imrec.in`, for output - `imrec.out`. You are not allowed to read or write any other files.

The first number on the input will always be a flag, signaling whether this is the first or second run - the number 0 or 1.

On the first run:

On the input file, **W** and **H** follow - the width and height of the original image.

On the next **H** lines there are **W** triplets of numbers with values between 0 and 255 - the colors (red, green, blue) of the pixels of the image.

On the output print 16 triplets of numbers (each number from 0 to 255) - the color palette you choose to use for the compression. On the next **H** lines print **W** numbers - the indices of the color palette for each pixel (from 0 to 15).

On the second run:

On the input file, the color palette you chose in the last run follows - 16 triplets of numbers. After that there are **W₁** and **H₁** - the width and height of the **altered** compressed image. On the next **H₁** lines there are **W₁** indices of the color used for each pixel (each from 0 to 15).

On the output, print the reconstructed color (3 numbers) for each pixel (**H₁** * **W₁** pixels).

Constraints

Original images are up to 500x500 in size.

After alternation at least 50% of the pixels in each dimension will be left for reconstruction.

20% of the pixels will be swapped with their neighbour, but only once.

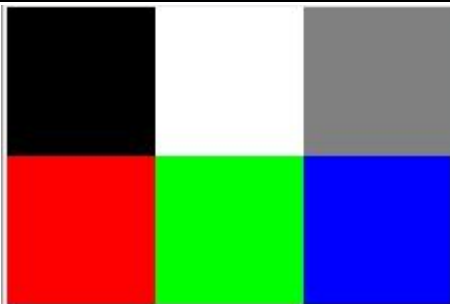
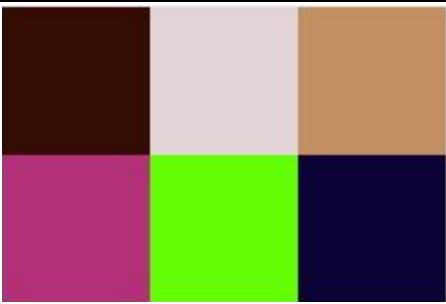
Pixels are neighbours when they share a side.

Time limit: 5 seconds for each run

Memory limit: 256 MB for each run

Example

There are more examples in the visualizer.

First run	
Input (imrec.in)	Output (imrec.out)
<pre>0 3 2 0 0 0 255 255 255 128 128 128 255 0 0 0 255 0 0 0 255</pre>	<pre>228 213 217 54 16 37 170 188 216 248 233 22 195 143 98 53 12 3 88 155 61 136 192 191 255 242 173 179 49 121 12 4 55 185 145 226 149 253 241 128 188 188 82 190 35 100 255 6 5 0 4 9 15 10</pre>
	
Second run	
Input (imrec.in)	Output (imrec.out)
<pre>1 228 213 217 54 16 37</pre>	<pre>179 49 121</pre>

170 188 216	
248 233 22	
195 143 98	
53 12 3	
88 155 61	
136 192 191	
255 242 173	
179 49 121	
12 4 55	
185 145 226	
149 253 241	
128 188 188	
82 190 35	
100 255 6	
1 1	
9	

After the alternation of the compressed image a single pixel is left, with color 9 from the palette (179 49 121). It differs with 22818 from the corresponding pixel (the red one, in the lower left corner). **Be aware, this example is very small and simple, your solution wouldn't be tested on such.**