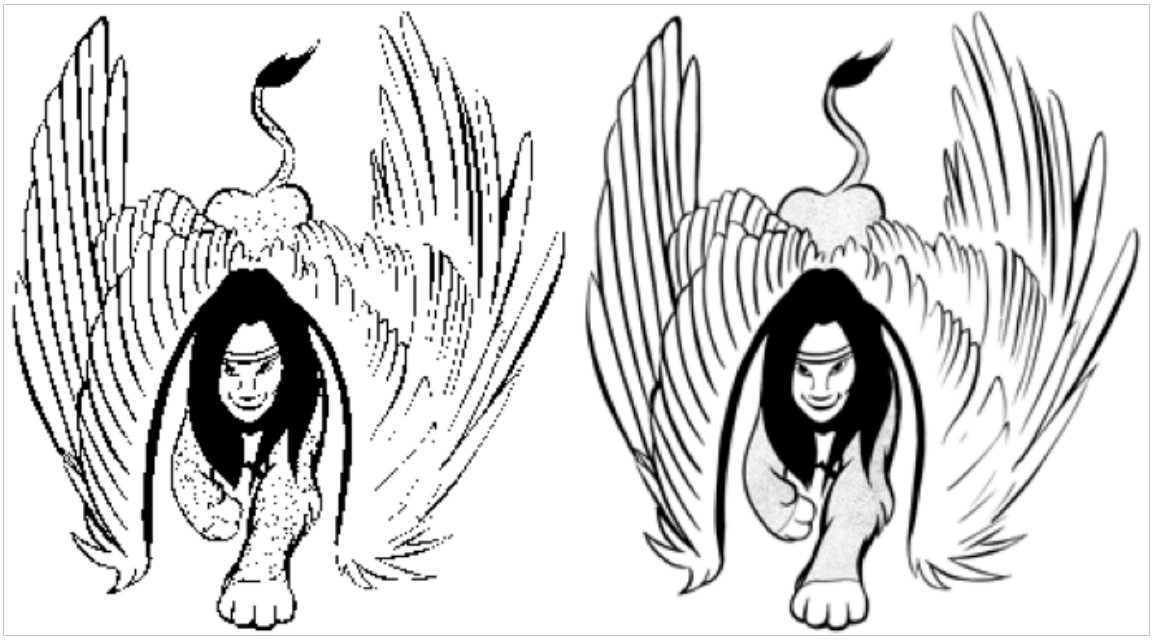
Anti-aliasing (SC-Camp 2014)

In computer graphics images can be seen as a two-dimensional matrix representing the color intensity of each pixels. Often, the quality of the rendered image may be improved using algorithms to cope with low resolution screens. One algorithm that improve graphics quality is called anti-aliasing. Anti-aliasing makes the boundaries of two regions smoother by approximating their colors. An example to illustrate how anti-aliasing improves image quality is presented below.



The left image show a sphinx that uses only black and white pixels. The image on the right is the output of the anti-aliasing algorithm. The anti-aliasing makes smoother color transitions adjusting each pixel based on the color of neighbor pixels. As we can notice with naked eye, the image quality is greatly improved after applying the anti-aliasing algorithm.

The Problem

Your mission is to make a parallel version of a simple anti-aliasing algorithm, very similar to the one described earlier. In this simple version each pixel has from 0 up to 255 tones of gray (8 bits per pixel). The problem is to compute a new image where each pixel is the mean of the pixel and its neighbors' pixels as follows:

$$P_{i,j} = \frac{\sum_{x=i-1}^{i+1} \sum_{y=j-1}^{j+1} P_{x,y}}{n}$$

A **non-optimized** sequential algorithm to solve this problem is:

```
begin
    for each pixel located at row i and column j
        s := sum of valid pixel neighbors (including the pixel i j)
        n := number of valid neighbors (including the pixel i j)
        newImage[i][j] := s /n
    end for
end
```

Image **before** anti-aliasing

200	0	125
0	0	0
10	0	125

To illustrate the algorithm, the example below shows a very small image, 3x3 pixels. The top table shows the input image (before anti-aliasing). The bottom image is the result image after applying the anti-aliasing algorithm.

Image **after** anti-aliasing

50	54	31
35	51	41
2	22	31

Input

A file that contais:

- two integers to indicate the size in columns (width) and rows (height) of the image, respectively;
- a series of integer representing the gray tone of each pixel.

Output

The value for each pixel.

Sample Input

```
3 3
200 0 125
0 0 0
10 0 125
```

Sample output

```
50 54 31
35 51 41
2 22 31
```