# Image processing course – homework #1 imageprocessinghaifau@gmail.com

In this homework you will be writing python functions of histogram operations.  
Read the whole document and the .py files before you start writing code.

**Getting started:**

* Open a new PyCharm project.
* Install new packages using pip (as shown in the tirgul#1 presentation)
* pip install numpy
* pip install opencv-python
* pip install matplotlib

numpy is used for calculations on arrays and matrices.  
from numpy, you’ll need to use: np.zeroes, np.ones, np.cumsum, np.matmul .

You are provided with two files **hw1\_123456789.py** and **hw1\_script.py**:  
The script file is not for submission; it shows an example of how to read images, call functions and display your results to check your solution.

You are also provided with Images to test your solution on.

* Insert the .py files and the images folder inside your project.

You can assume:

* Input images are completely safe (not empty or corrupted).  
  you don’t need to check the input to the function

**Functions:**

You can assume: Input images are completely safe (not empty or corrupted).  
you don’t need to check the input to the functions.

1)   **histImage(im)**  
            This function calculates the histogram of a given image.

**Input**:      im - a grayscale image in the range [0..255]

**Output**:   h - image histogram. An array 1x256 such that entry i contains  
                           the number of pixels in the image having grayscale value i.

**Method**:  Counts the number of pixels with grayscale value i in the range [0 .. 255].   
                           You can loop over the image pixels incrementing the appropriate histogram bin or you can loop over the grayscale values (0..255).   
  
Note: Do not use built in functions to calculate the histogram.

**2)   nhistImage(im)**  
          This function builds the normalized histogram of image im.

**Input**:      im - a grayscale image matrix in the range [0..255]  
**Output**:    nh - normalized image histogram = an array 1x256 such that entry i contains the  
                           proportion of pixels in the image having grayscale value i  (i.e. the number of pixels of value i divided by the total number of pixels in the image).

**Method**:  Counts the proportion of pixels with grayscale value i in the range [0 .. 255].   
                 Use function histHimage inside this function.

**3)   ahistImage(im)**  
          This function builds the accumulated histogram of image im.

**Input**:      im - a grayscale image matrix in the range [0..255]

**Output**:    ah – accumulated image histogram = a vector 256x1 such that entry i contains the number pixels in the image having grayscale value **i or less**.

**Method**:  You don’t need to loop over the image inside this function. Use function histHimage to calculate histogram of image, and from it the accumulated.

**4)   calcHistStat(h)**  
          This function calculates statistics of the input histogram: Mean, Variance.

**Input**:      h - image histogram. An array 256x1.

**Output**:    m, v

m – mean pixel values.  
             v - variance of pixel values.

**Method**:  Calculate m and v using 1 line of code each!! (Matrix multiplication)

Important note: you can use the functions np.mean(im) , np.var(image) in the script to compare and make sure your answer is correct. **DO NOT** use them to implement the function itself.  
5)     **mapImage (im,tm)**  
            This function maps image grayscale according to given tone map. Returns new image.

**Input**:      im - a grayscale image matrix in the range [0..255]  
                 tm – tone map = a 256x1 vector defining a mapping.

**Output**:     nim – the new grayscale image (same size as im).

**Method**:  Map gray value i in image to new value given in tm[i].   
You can implement this function using 1 line of code!  
Or you can loop over grayscales values [0..255], not over image pixels.

Note: make sure all values (in the new image - after applying the tone mapping) that are greater than 255 are reduced to 255. All values smaller than 0 to be 0.

6)         **function    histEqualization(im1)**  
            Uses **2 pointer algorithm** (from the lectures) to calculate the tone mapping necessary for histogram equalization

**Input**:      im1 - grayscale images - matrix in the range [0..255]                             
**Output**:     tm – tone map = a 1x256 vector defining a mapping.  
**Method**:  Calculates the tone map tm required for histogram equalization using the 2-pointer algorithm.   
use ahistImage to get accumulative histogram.  
the 2-pointer algorithm work on two histograms - what is our goal histogram?  
You can use the function np.cumsum(goal\_histogram) to get the accumulated goal histogram.

**Submission**

Please submit one .py file with the functions implemented

Name the file **hw1\_123456789.py** (or in case of pair: **hw1\_123456789\_987654321.py**)

**(Replace 123456789, 987654321 with your ids)**

# Good luck!