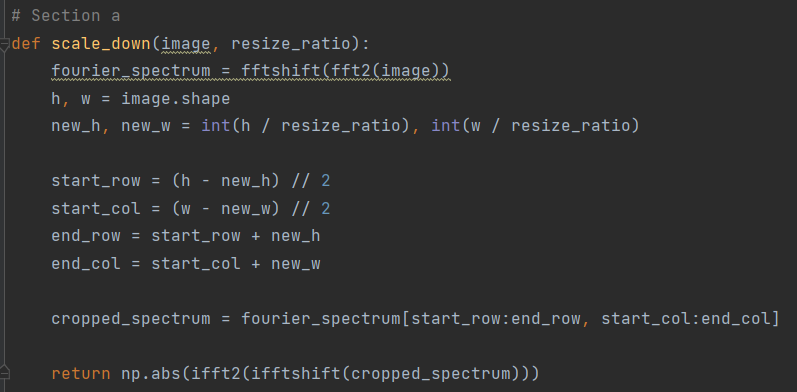
Image Processing HW\_5

Student -1- : Mostufa Jbareen (212955587)מוסטפא גבארין

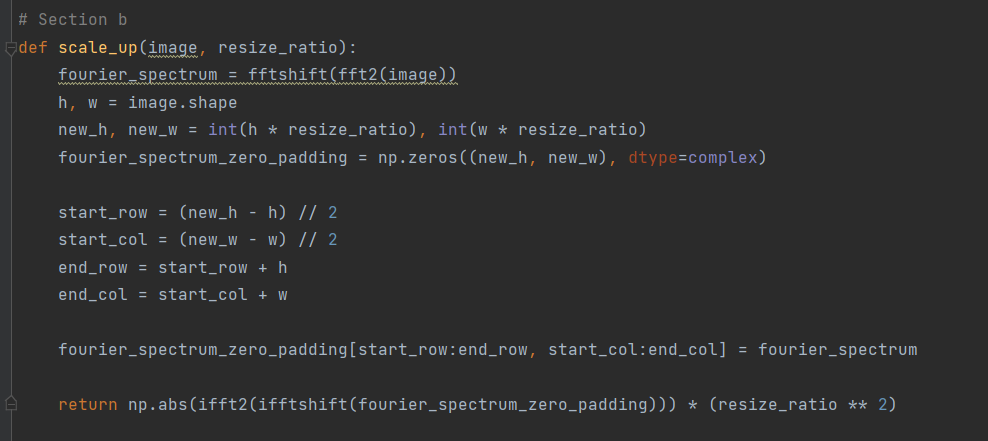
Student -2- : Mohammed Egbaria (318710761) מוחמד אגבאריה

**Problem 1:**

**Section a:**

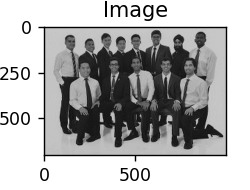
Implementing scale\_down function by applying our knowledge on Fourier, we know that a scaled down image by ratio is the image smaller in that ratio and has and without the high frequencies in the same ratio, so we take the crop the Fourier transform by the resize\_ratio starting from the lower frequencies, and then we inverse the cropped window to get the sacled\_down image.

**Section b:**

The same thing we did in scale\_down, butt this time we need to strentgth the freuquencies because now we have a bigger image, and we do that in the last line

Example:

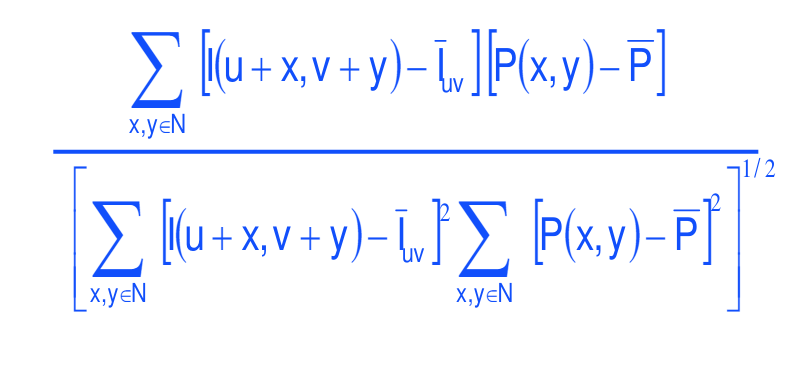
The crew image scaled up with resize\_ratio = 4:



We can see now the image contains more pixels than the original and in the same quality

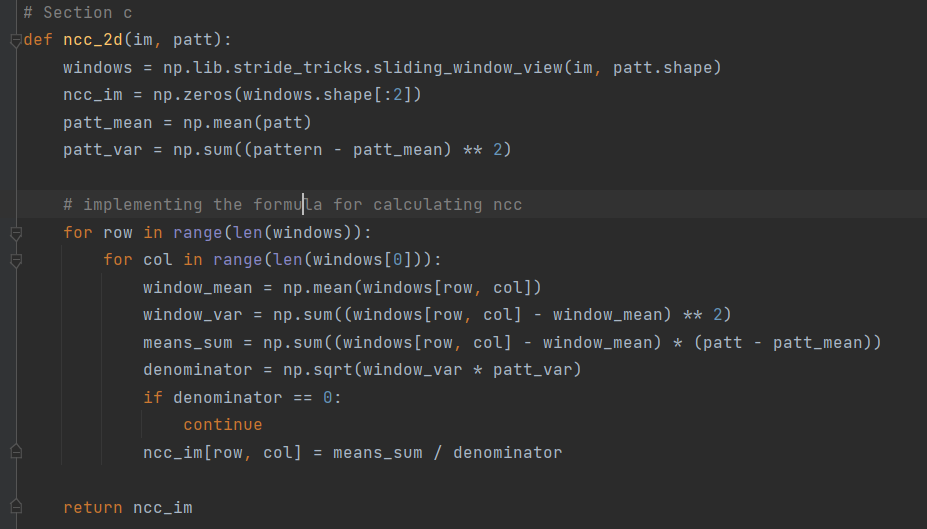
**Section c:**

This function calculates the ncc (Nomalized Cross Correlation) between an image and a pattern, using sliding windows, mean and variance. This fuction used in “find the pattern in the image problem”.

The following implementation is based on this formula:

Where:  
P – pattern  
I – Image  
I— -image mean

P— -pattern mean

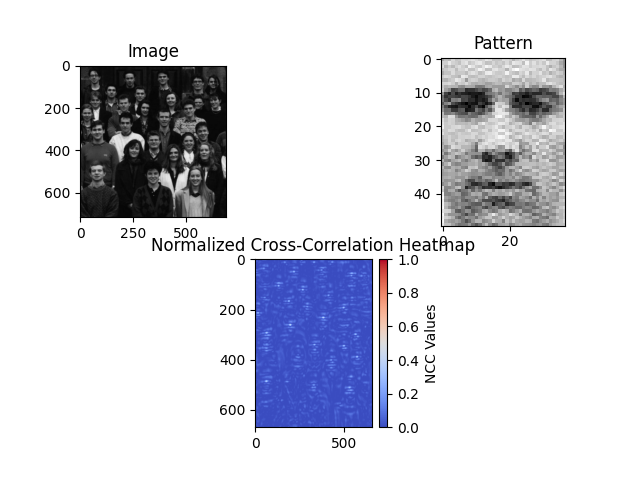


**Section d:**

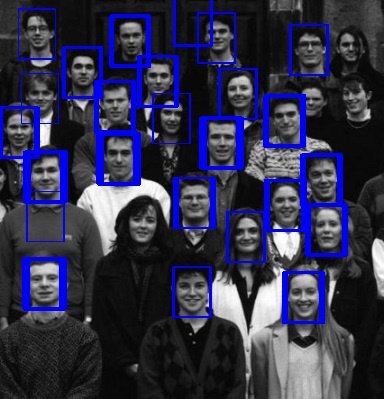
Students image:

Display:

image\_scale\_ratio = 1.8  
pattern\_scale\_ratio = 1  
threshold = 0.46



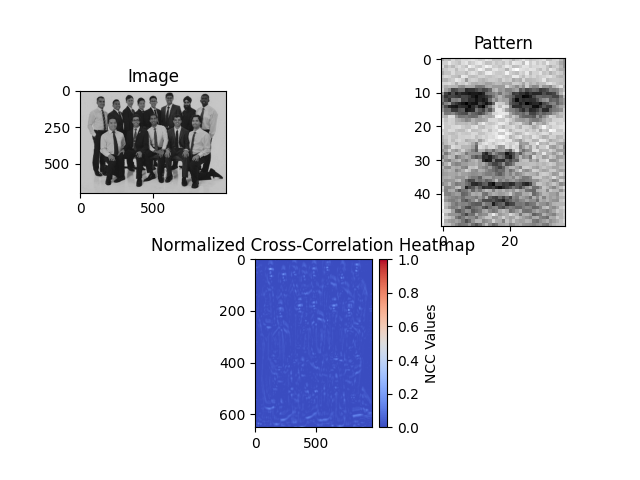
Result with false positives:



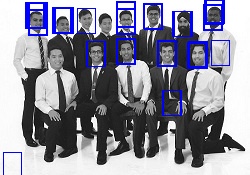
Thecrew image:

Display:

image\_scale\_ratio = 4  
pattern\_scale\_ratio = 1  
threshold = 0.37



Result with false positives:



**Section e:**

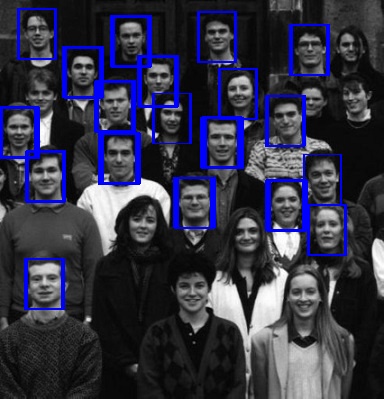
**Our best Results**

After search and trial and error, here are our best results

Students:

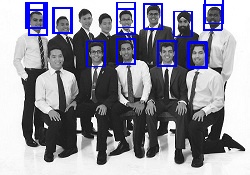
we chose these params:

image\_scale\_ratio = 2  
pattern\_scale\_ratio = 1  
threshold = 0.55

and the result is:  


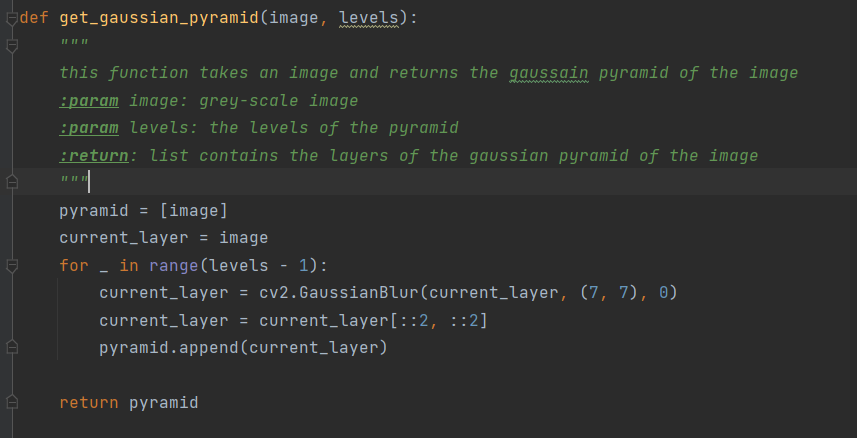
thecrew:  
we chose these params:

image\_scale\_ratio = 4  
pattern\_scale\_ratio = 1  
threshold = 0.4



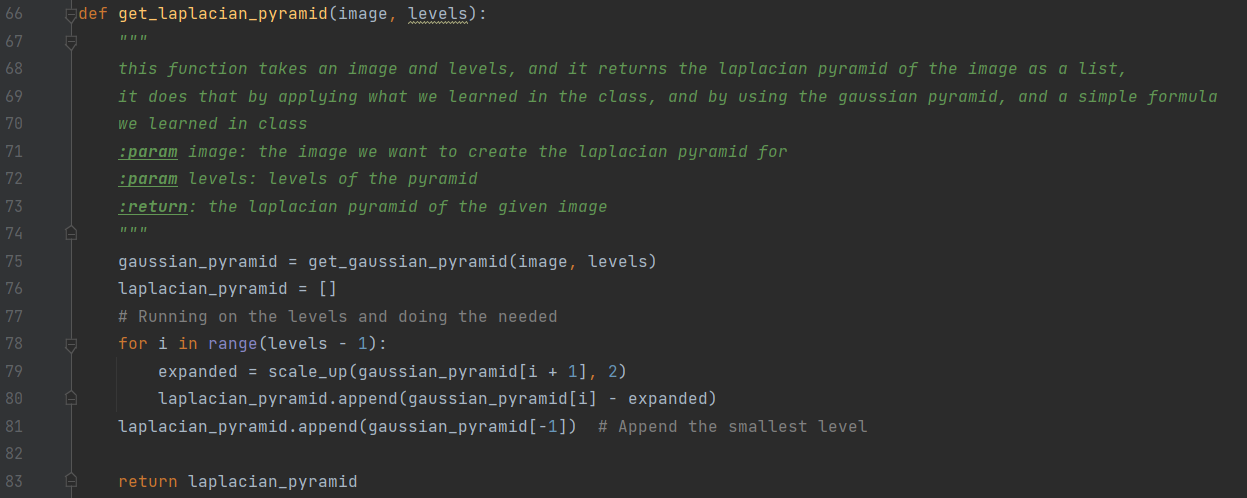
Problem 2:

First I want to mention that I implemented these aid functions in addition to the functions required in the question:

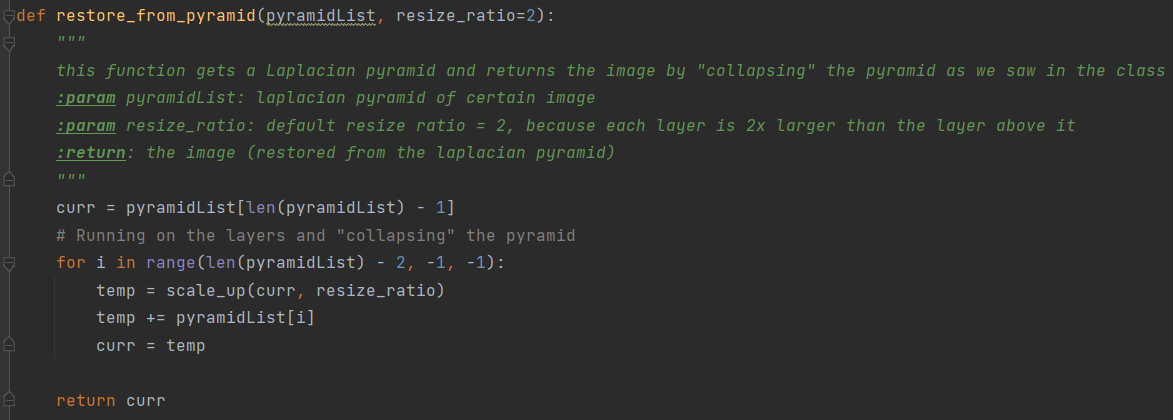


**And i used scale\_down() and scale\_up() from Question 1**

**Section a:**

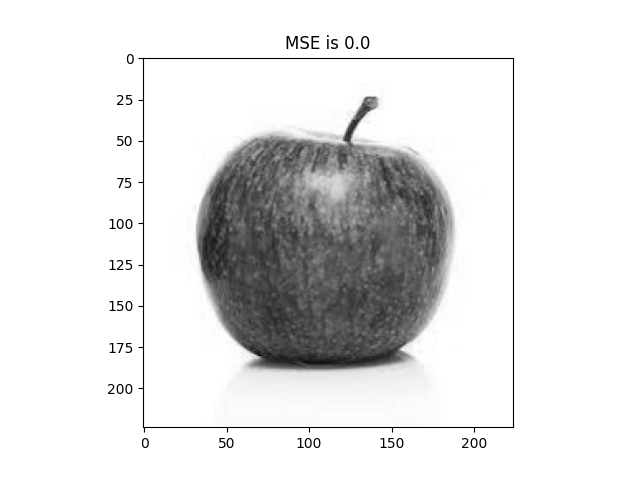
****

**Section b:**

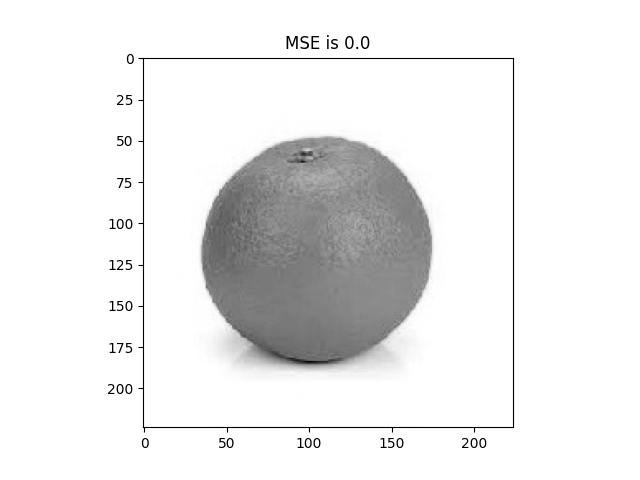


By using the given function validate\_operation, we created Laplacian pyramids for both orange and apple and we restored it and we got this results:

Apple:

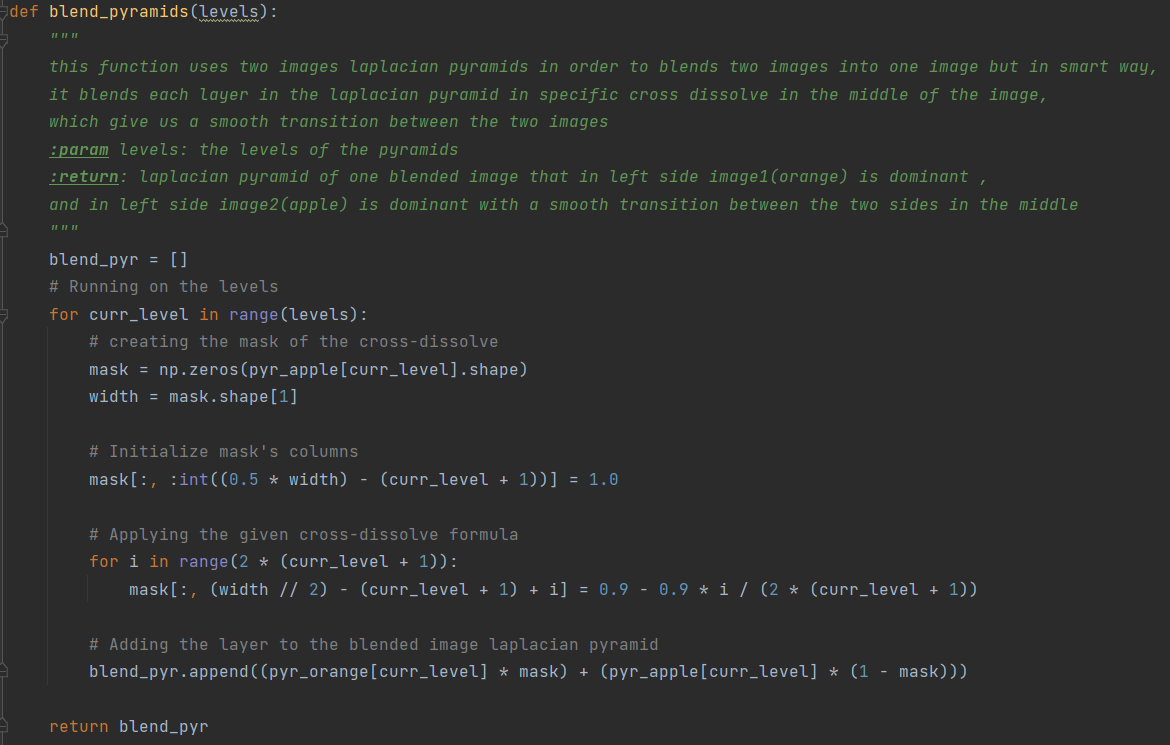


Orange:



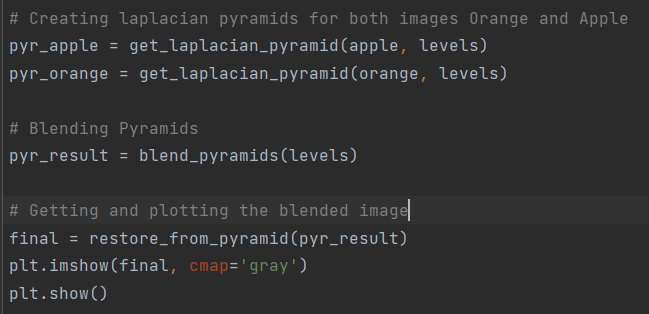
Less than 1, so NICE!

**Section c:**



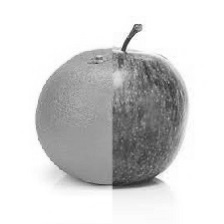
**Section d:**

We created Laplacian pyramids for both images Apple and Orange, and then we blended the two images using our blend function:



But before that we needed to choose the best level value that gives the best image we can get from blending the two images, after trying all the possible levels, we found out that level=3 is the perfect match in the trade-off between the smoothens in the cross-dissolve and between the “showing” that this is blended image (there is dominant part in each side has its own colors)

Level = 1:



Level = 2:



**Level = 3: our best result**



Level = 4:



Level = 5



in levels = 1\ levels = 2, we clearly can see the line in the middle that separates the two images, the transition between the images is not smooth and it is not good.

While in levels = 4\ levels = 5, we can see also the fainting in the colors of each side, which is also not good.

And finally in levels=3 we find the best results, we have a good blending and also we have great colors for each side.

**Bonus Question:**

**Section a:**

I completed this section in the code file in many different places

**Section b:**

I used the following canny implementation to find edges in the image:

min\_edge\_threshold, max\_edge\_threshold = 100, 200  
edge\_image = cv2.Canny(edge\_image, min\_edge\_threshold,  
 max\_edge\_threshold) # Apply edge detector with min\_edge\_threshold, max\_edge\_threshold

**Section c:**

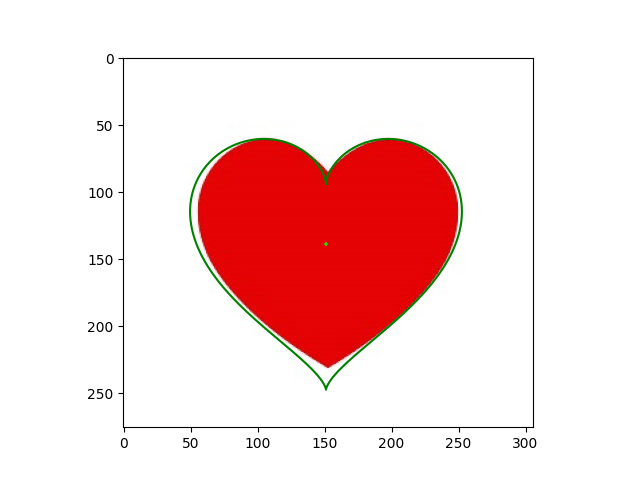
After many trail and error I found these parameters that led me to the best result I had:

**simple:**

Parmeters:

r\_min = 6.5  
r\_max = 7.5  
bin\_threshold = 0.18

Result

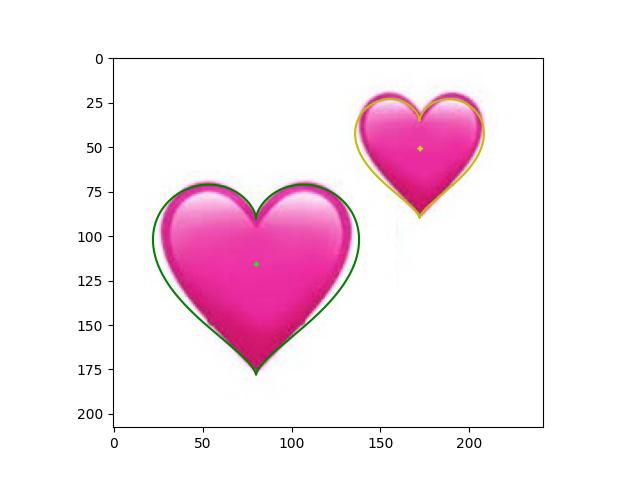


**med:**

Parameters:

r\_min = 2  
r\_max = 5  
bin\_threshold = 0.3

Result:

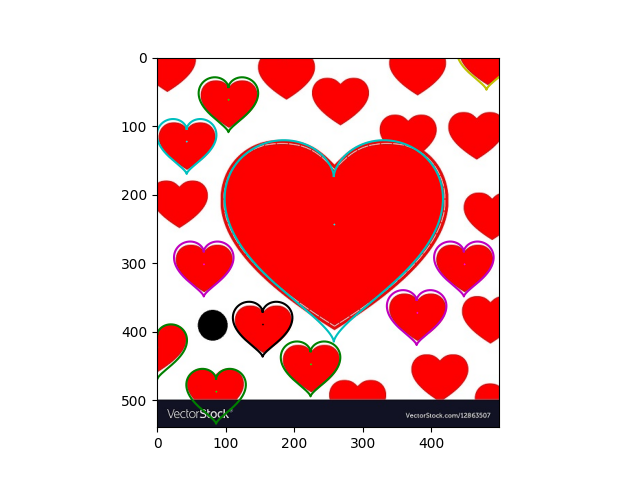


**hard:**

Parameters:

r\_min = 3  
r\_max = 12  
bin\_threshold = 0.26

Result:



We can see that we succeeded to find all the hearts in simple and med image, but we have a problem to do that in the hard image, and that is because we can see that in simple and med image we only have hearts and few of them (simple contains 1 and med contains 2) while in the hard image we not only have a much more hearts but also we have also different shapes in the image itself, like the black circle and the black rectangle below, these factors makes it hard to detect all the hearts in the image.