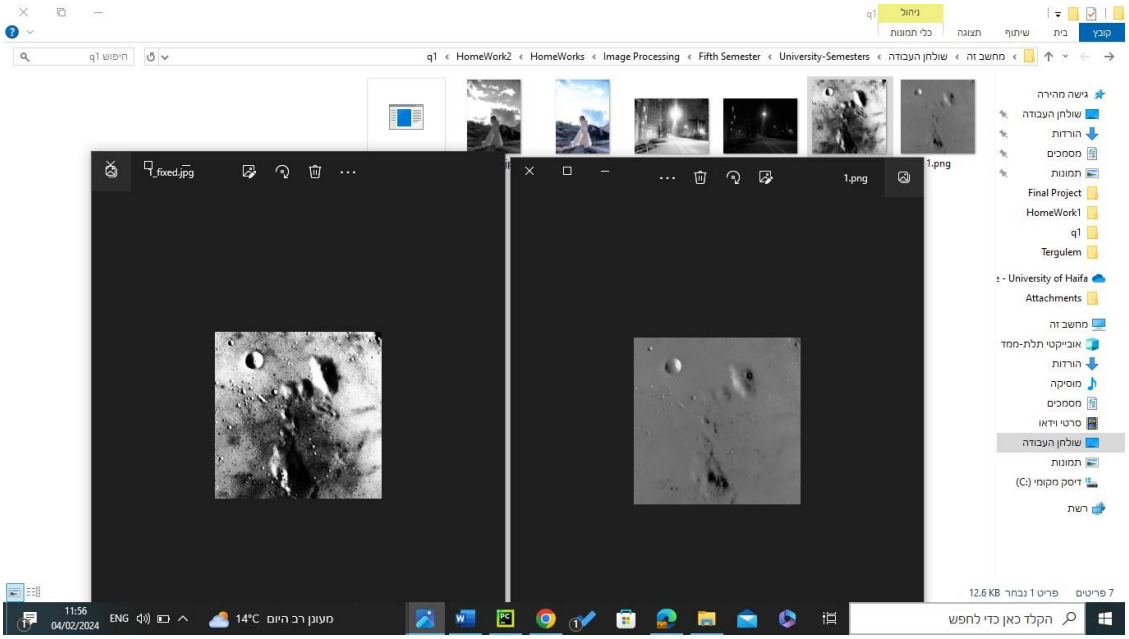
Home Work 2 – Image Processing

Question 1)

Histogram Equalization:

First I tried using Histogram equalization and this is what happened :

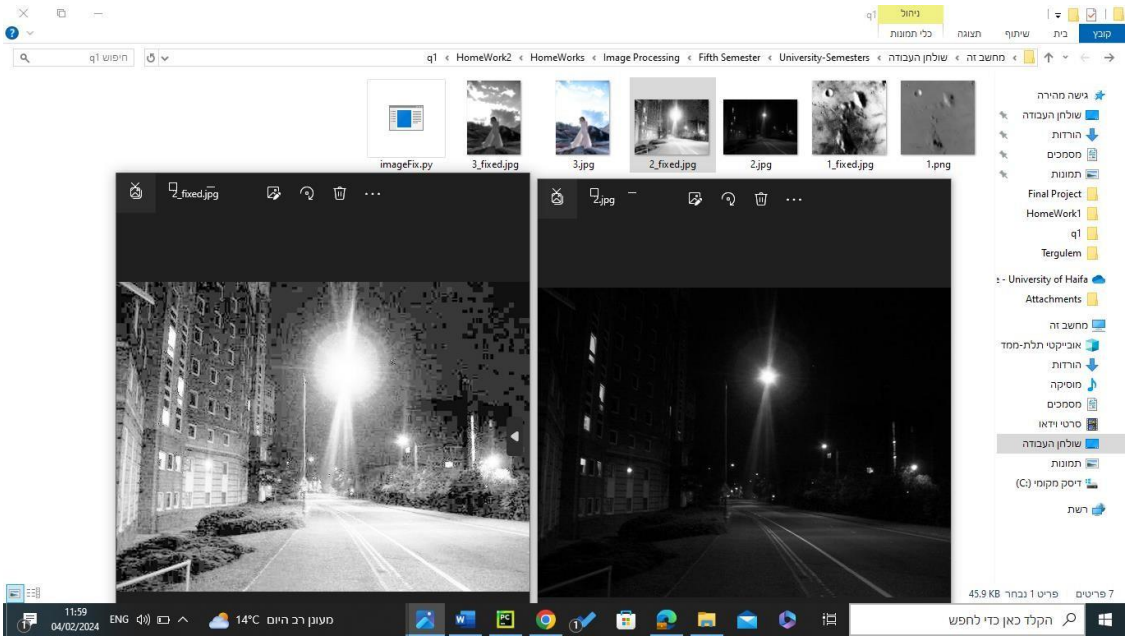
In image 1: we can see the left side(fixed) is a bit more defined than the right side. This is because by performing histogram equalization, the overall contrast of the image is enhanced, and details that were previously hidden due to poor contrast may become more visible.



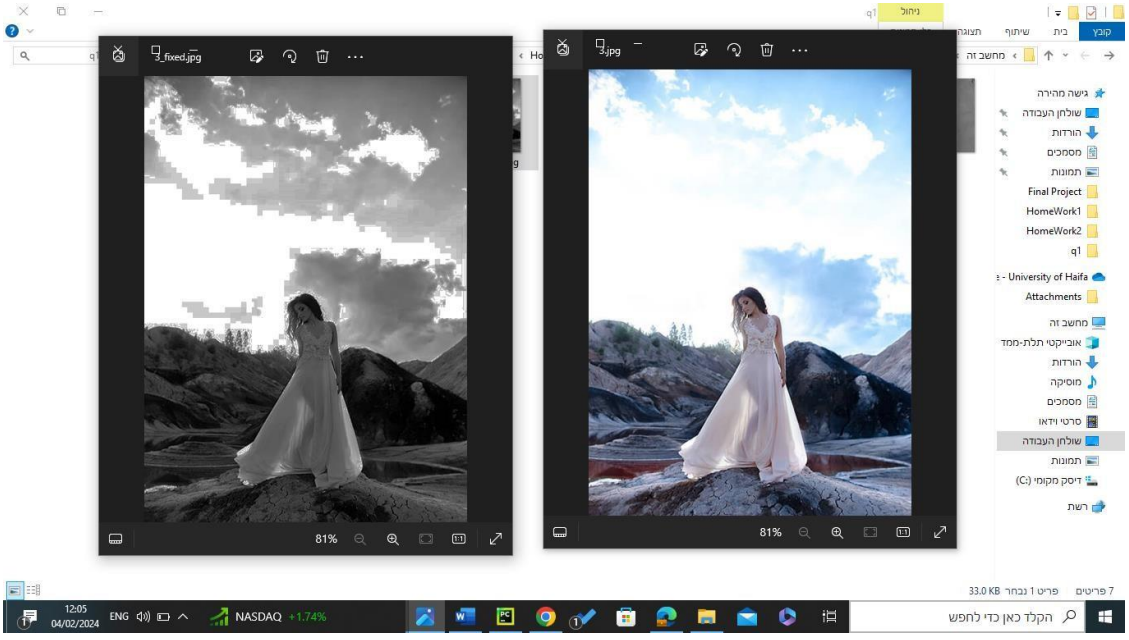
In image 2:

In this image however the result of histogram equalization was bad and very blurry, this is because histogram equalization works best when an image has a wide range of pixel intensities. In a dark street image with few lights, there might be limited contrast between the illuminated areas and the dark regions, making it challenging for histogram equalization to enhance contrast effectively.

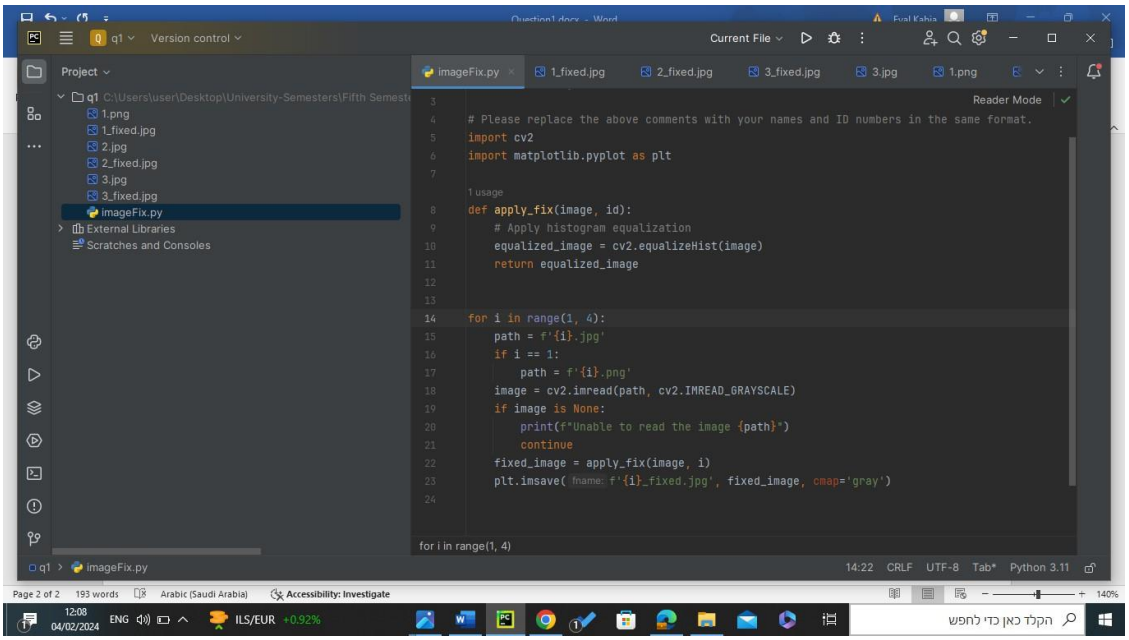
Also 1 factor that may cause this is amplification of noise, since dark images often have a higher level of noise, which can be amplified during histogram equalization. This can lead to an increase in image brightness and blurriness, as you observed.



In image 3: its very similar to image 1 and 2



This is the code I used for histogram equalization:



Brightness and Contrast Stretching:

Now I am trying to do Brightness and contrast stretching, to do this I have to check a few different alpha(between 1.0 to 3.0) and beta(between 0-100) parameters to see which are the best for my specific image:

Alpha=1.5, Beta=50:

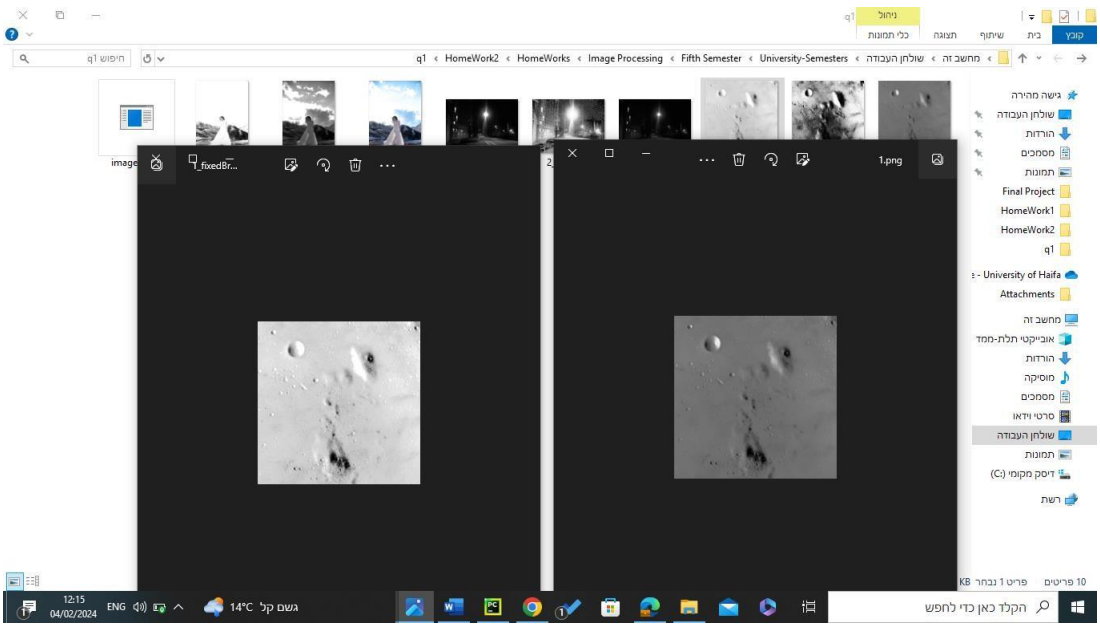
Images 1 and 2 have been enhanced and they look better and more defined, however image 3 seems to have lost some features and hasn't been enhanced but rather the opposite.

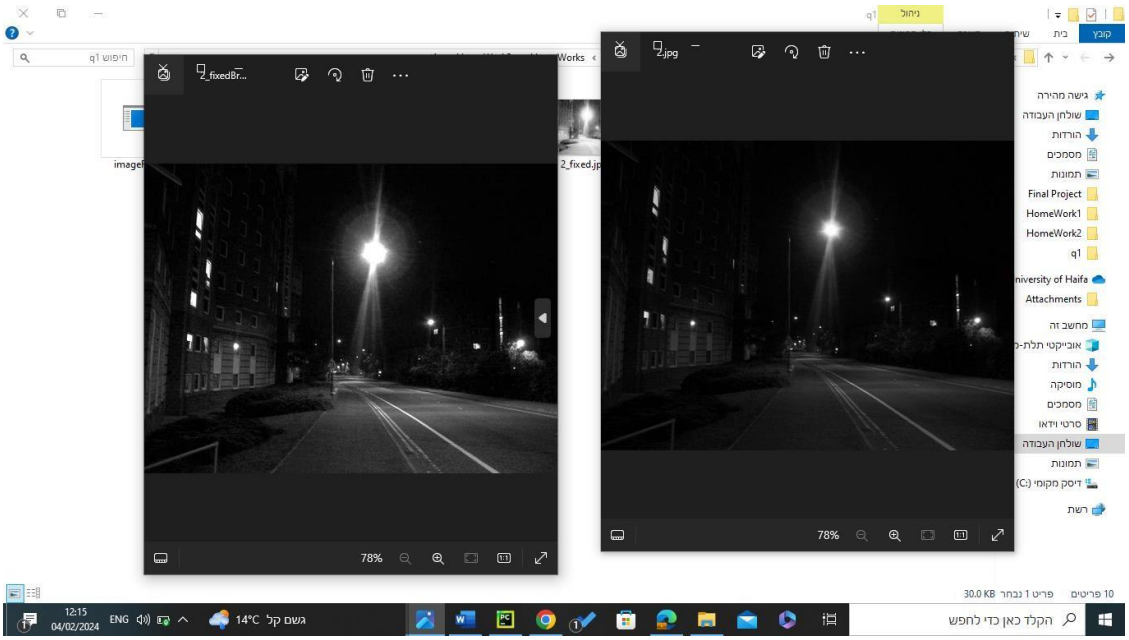
This can be attributed to a few reasons: **Clipping**: When adjusting .1 the contrast and brightness, if the dynamic range of the pixel values exceeds the limits of the data type used to represent the image (e.g., 8-bit or 16-bit), clipping can occur. Clipping results in the loss of detail in the highlights or shadows, leading to information loss in bright or dark regions of the image.

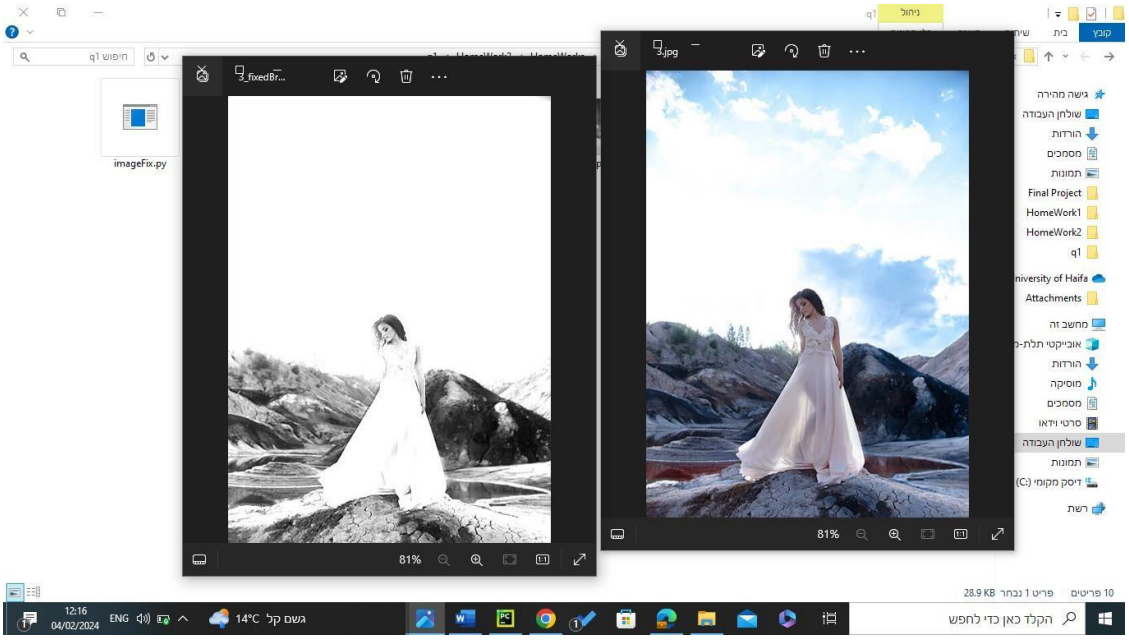
Saturation: Bright areas of the image, such as the sky, might .2 become overexposed and saturated if the contrast and brightness adjustments are too aggressive. This can cause the colors to appear washed out and lose detail, resulting in a loss of distinction between different shades of blue in the sky and clouds.

3. **Inadequate Dynamic Range**: Images with very high brightness levels or wide dynamic range may not be effectively enhanced using simple contrast and brightness adjustments alone. These images may require more sophisticated techniques, such as HDR (High Dynamic Range) imaging, to preserve details in both bright and dark regions while enhancing contrast.

4. **Lack of Local Contrast Enhancement**: Contrast and brightness stretching operate on the global pixel intensity values of the entire image. However, if there are regions within the image that require different levels of enhancement (e.g., bright sky versus darker foreground), a global adjustment may not be sufficient to preserve details in all areas.

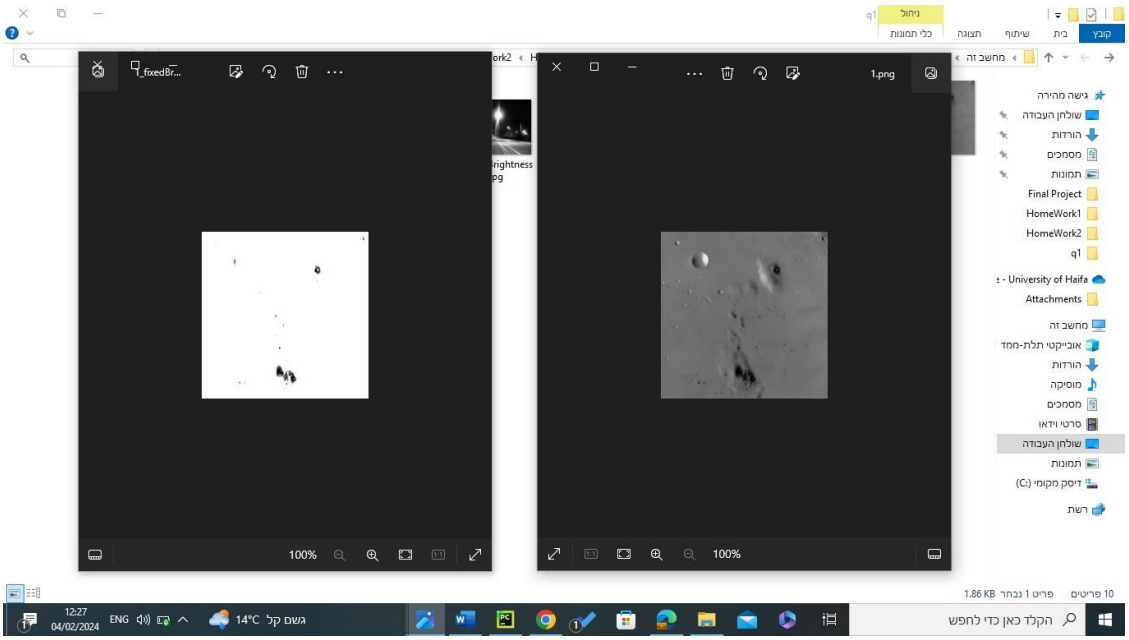


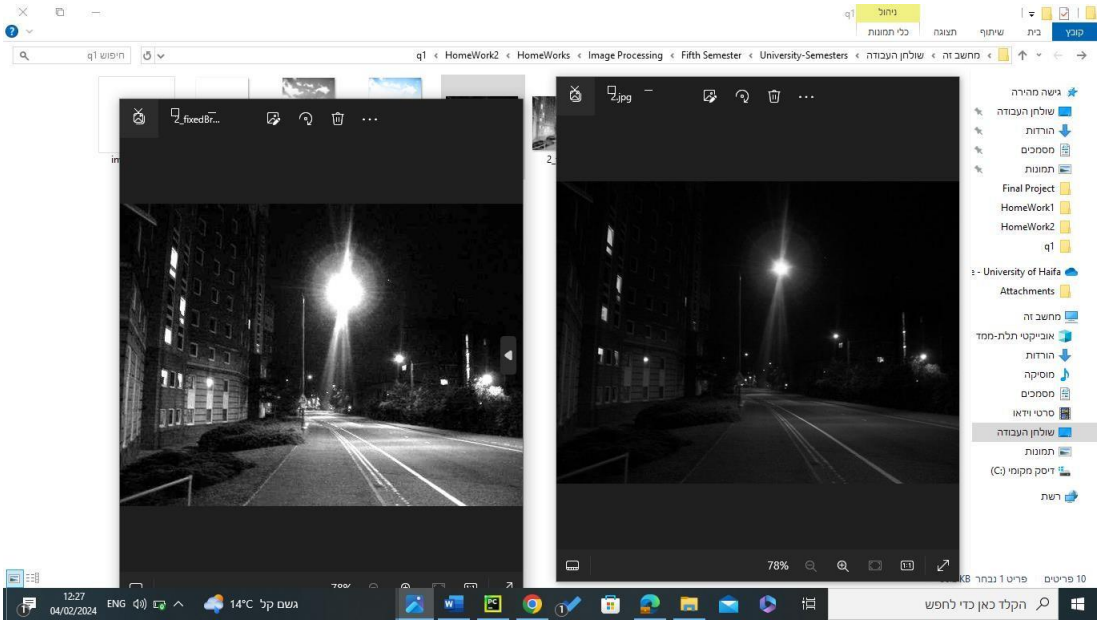


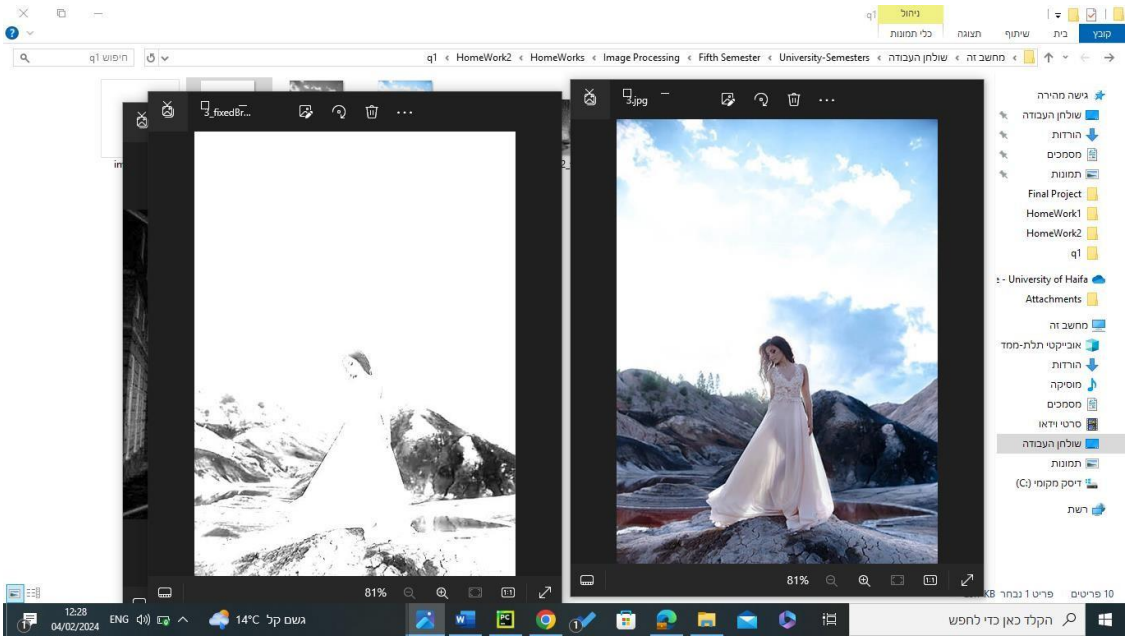


Alpha = 2.5, Beta = 90:

In here we can see that going aggressive with alpha and beta we lose definition in images 1 and 3 , but it wont show much on picture 2


1.png
100%
100%
Final Project
HomeWork1
HomeWork2
q1
Attachments
הקלד כאן כדי לחפש

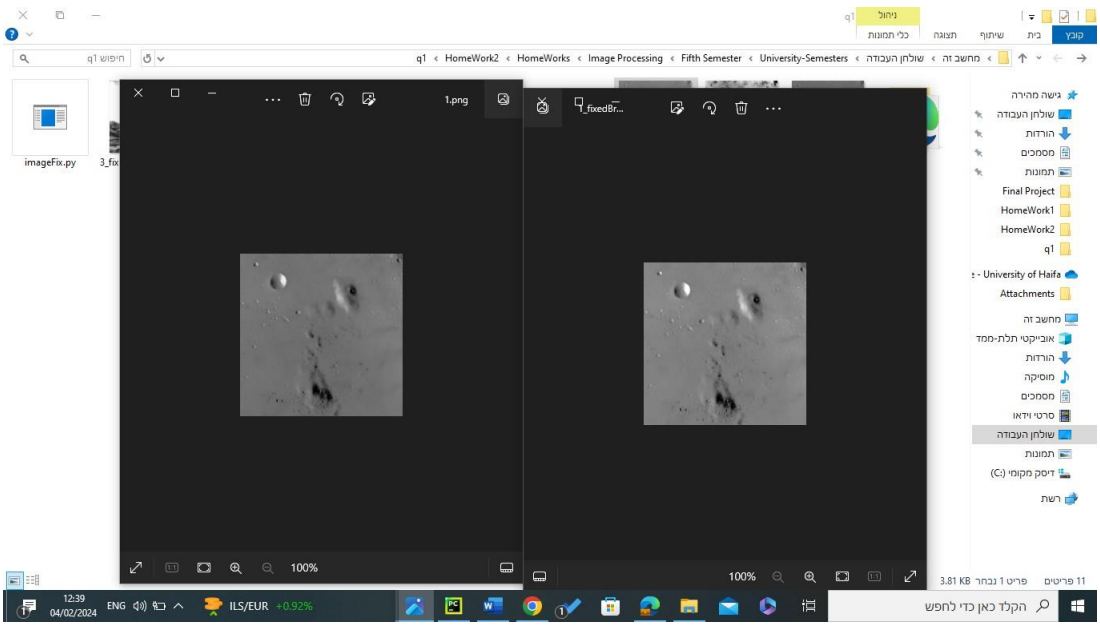

2.jpg
78%
HomeWork2
HomeWorks
Image Processing
Fifth Semester
University-Semesters
Final Project
HomeWork1
HomeWork2
q1
Attachments
הקלד כאן כדי לחפש

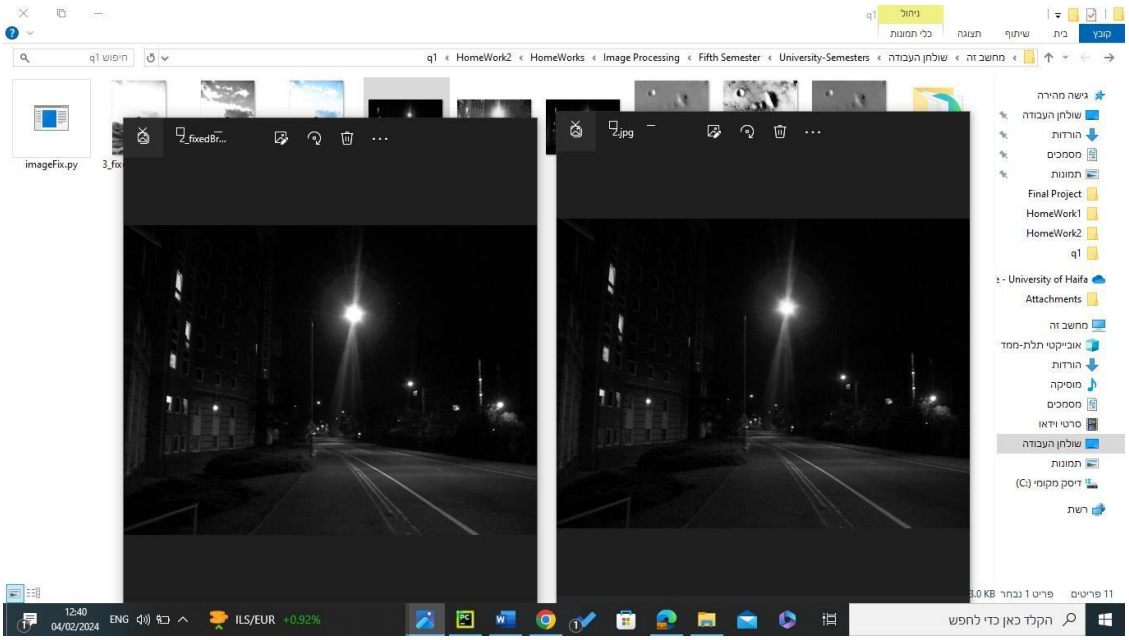


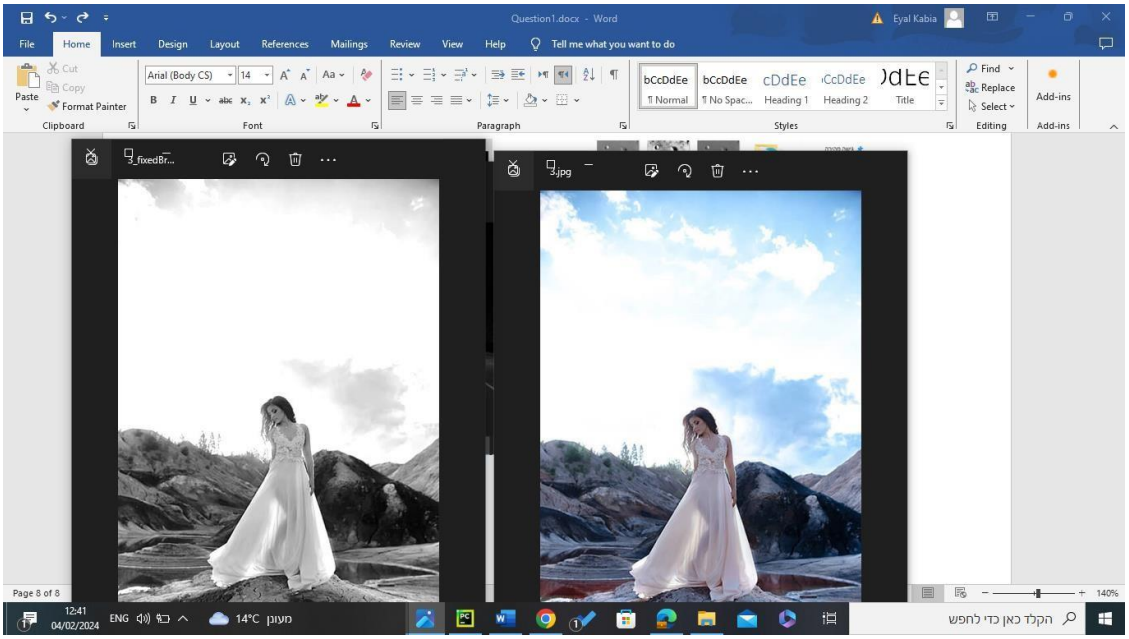
Now lets try the alpha and beta on the low end of the spectrum:

Alpha=1.1, Beta = 10:

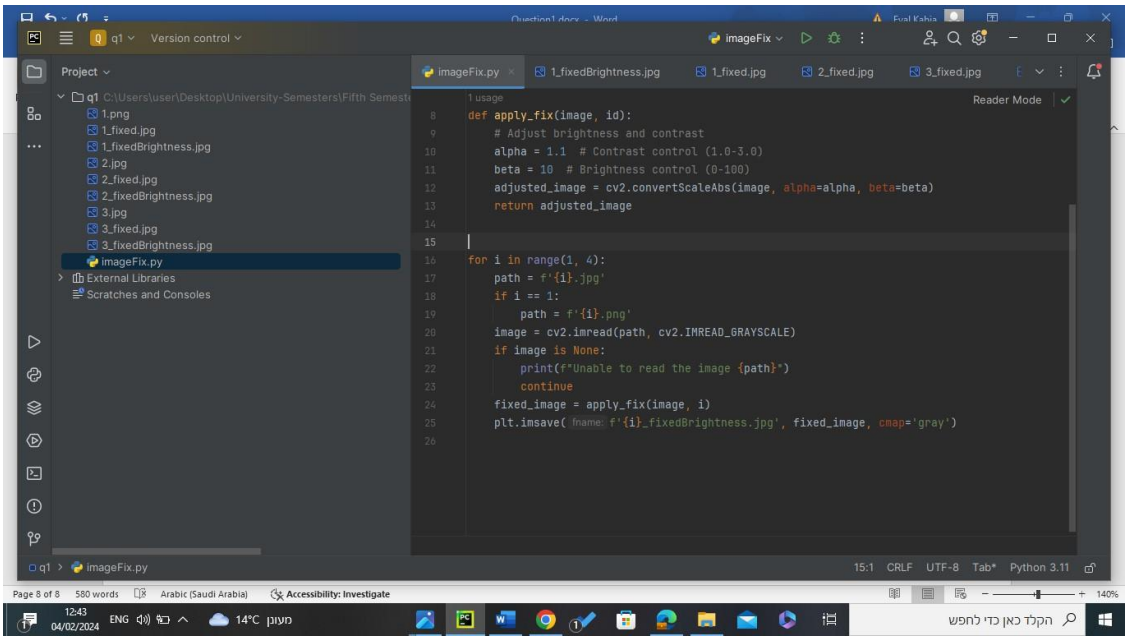
As we can see down below this didn’t really change much in the pictures(the gray scale in image 3 does not concern us since it was given in the skeleton code)







This is the relevant code section:

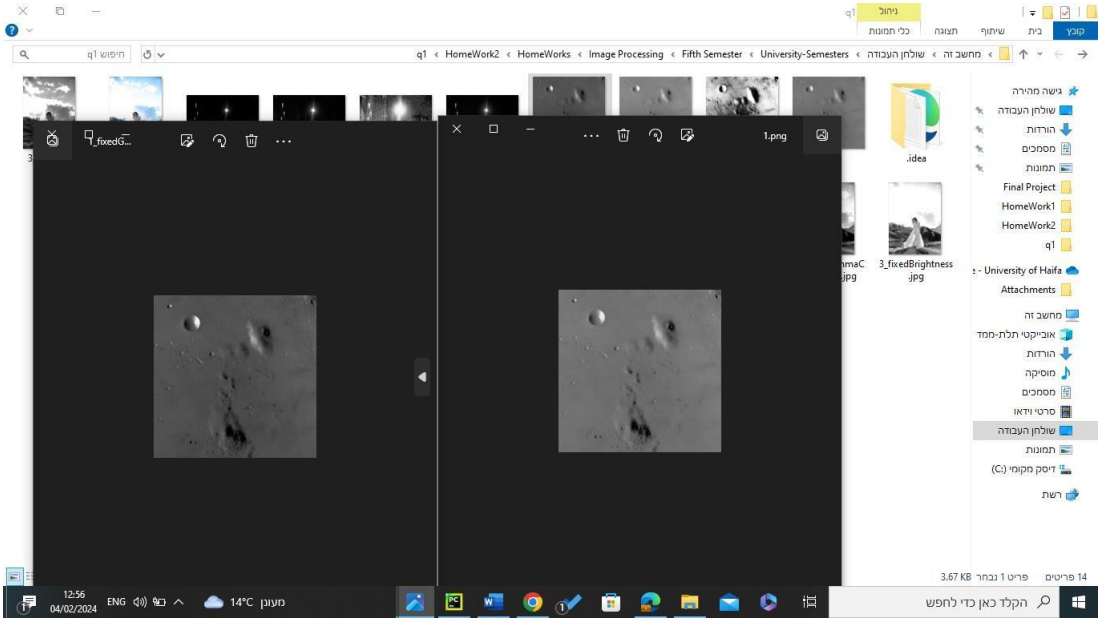


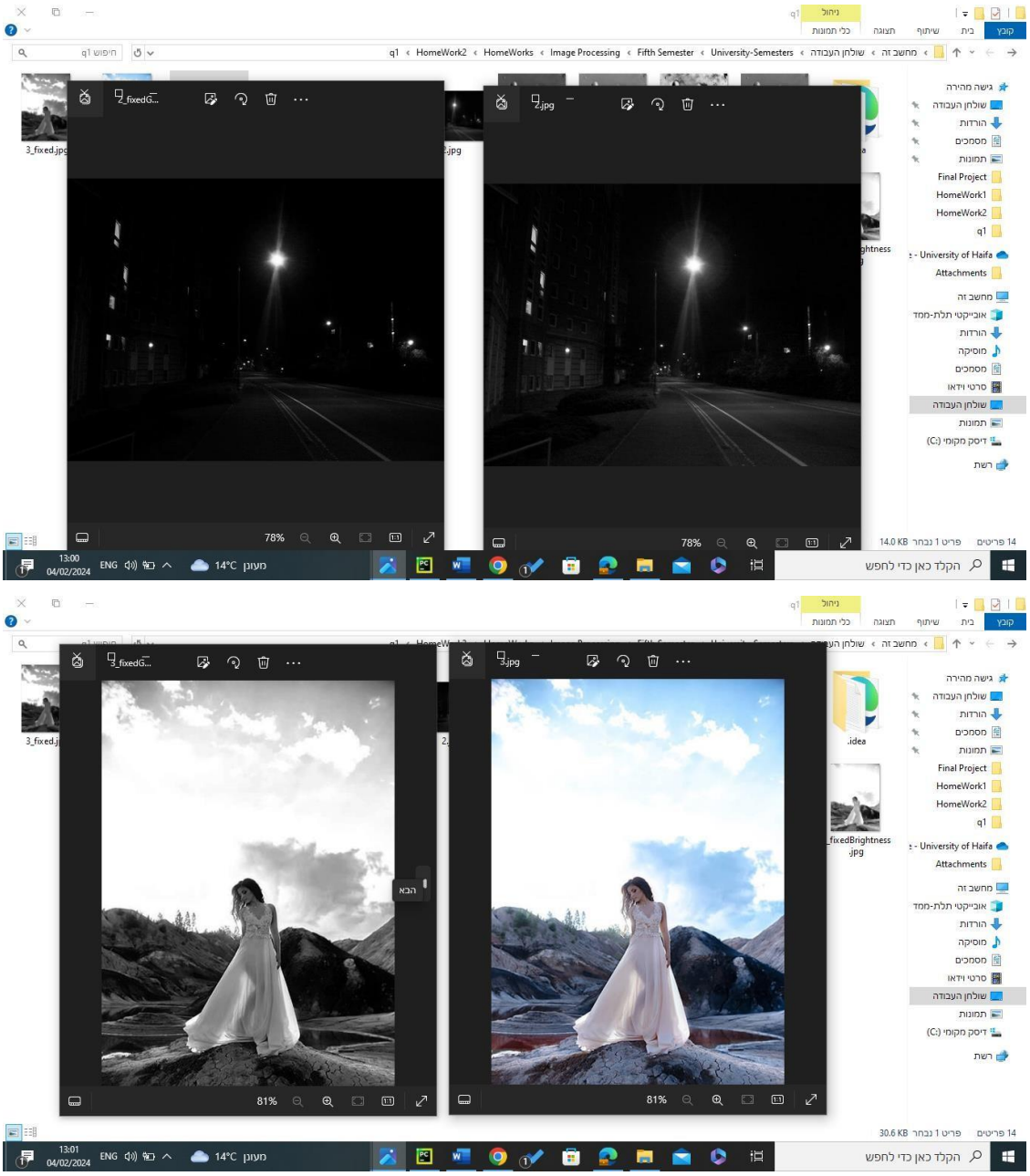
Gamma Correction:

In this correction we experiment with different gamma values: as we said in the tutorial if gamma value is bigger than 1 the image becomes darker and if gamma is between 0 and 1 the image becomes brighter, time to test out that theory:

Gamma=1.5:

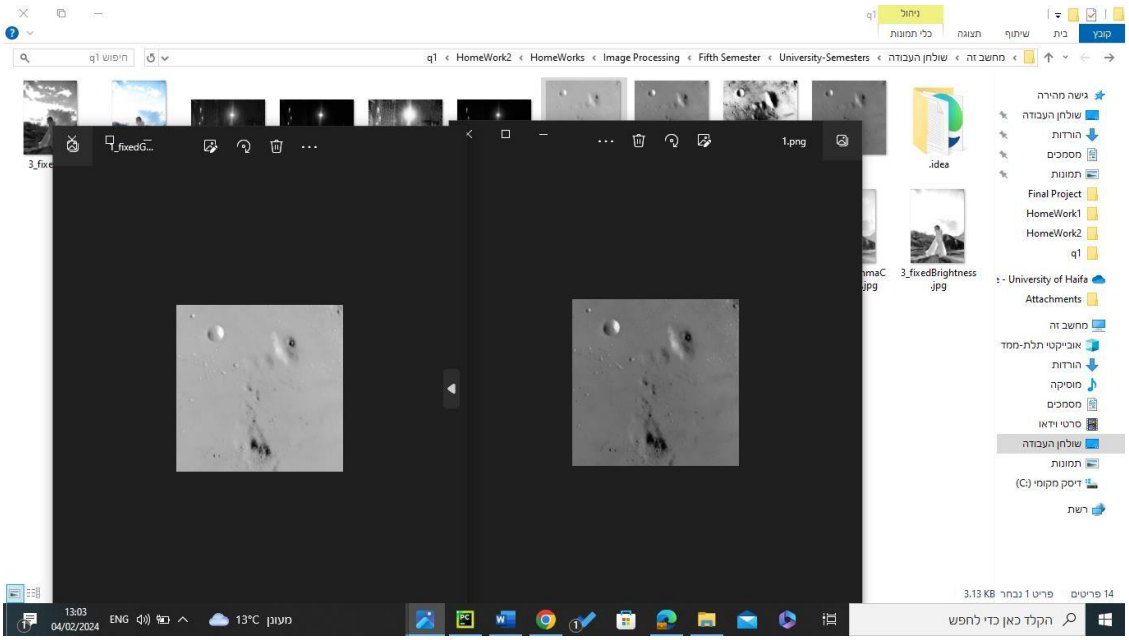
As we can see the images did in fact get darker and a little more defined but we didn’t enhance it much

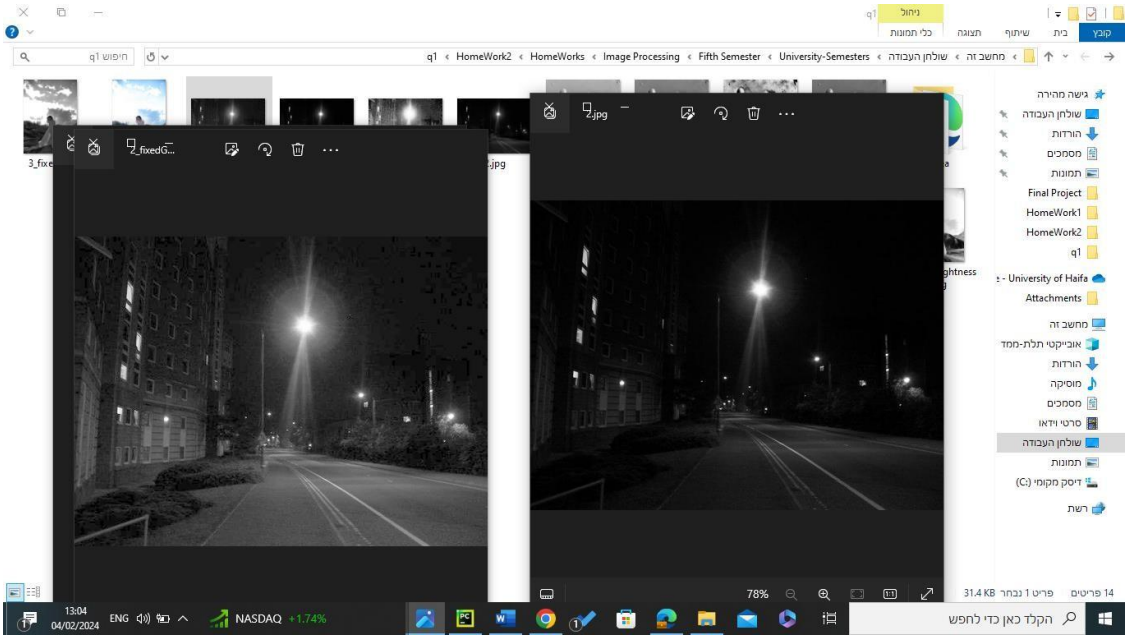




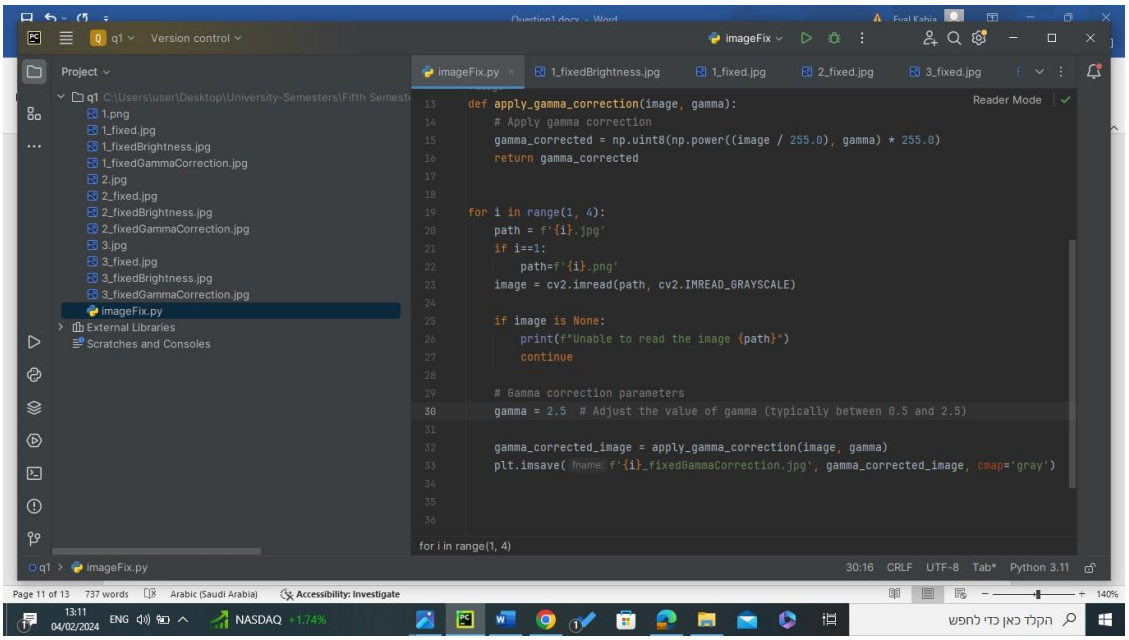
Gamma = 0.5:

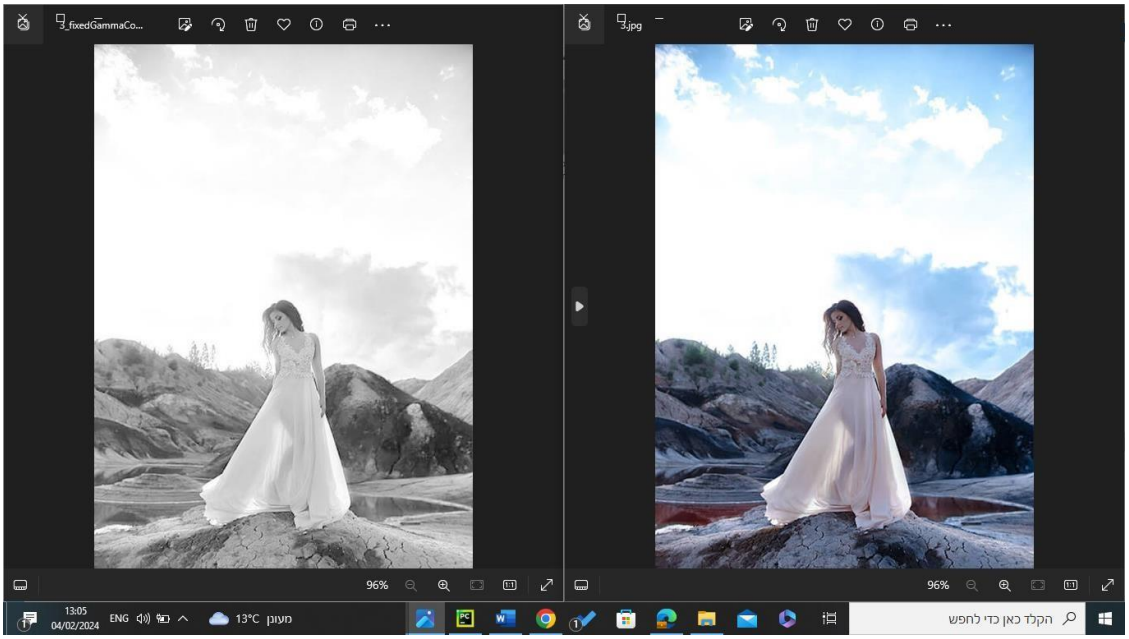
In here we can see the image got brighter as expected, however in image 2 we can see that we added some blurriness and weird noises in the background





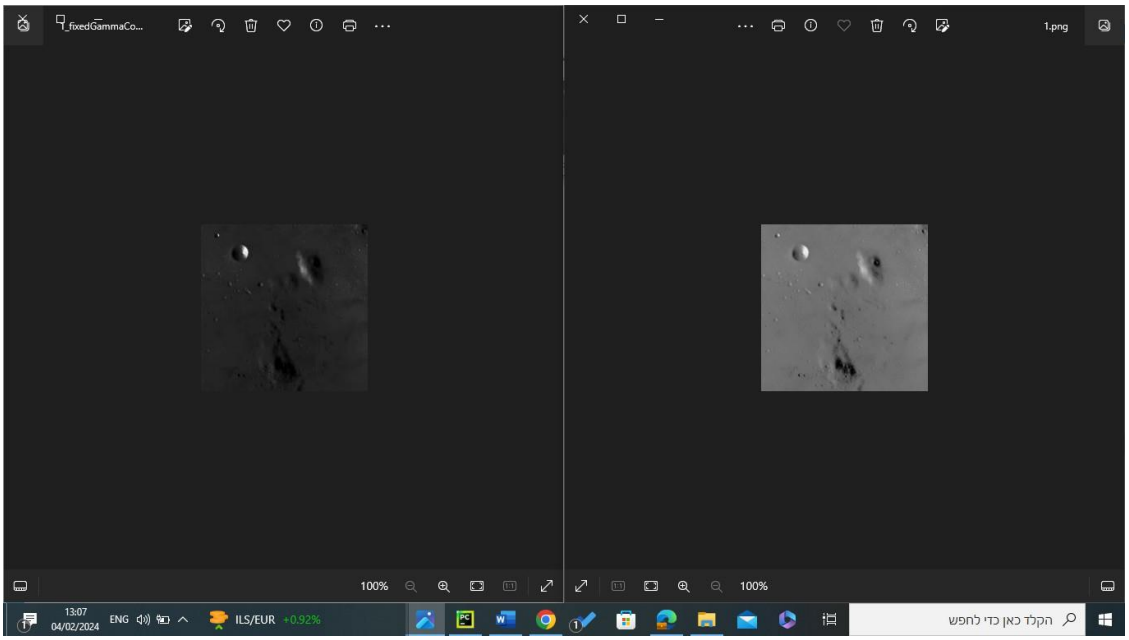
This is the relevant code section for this correction:

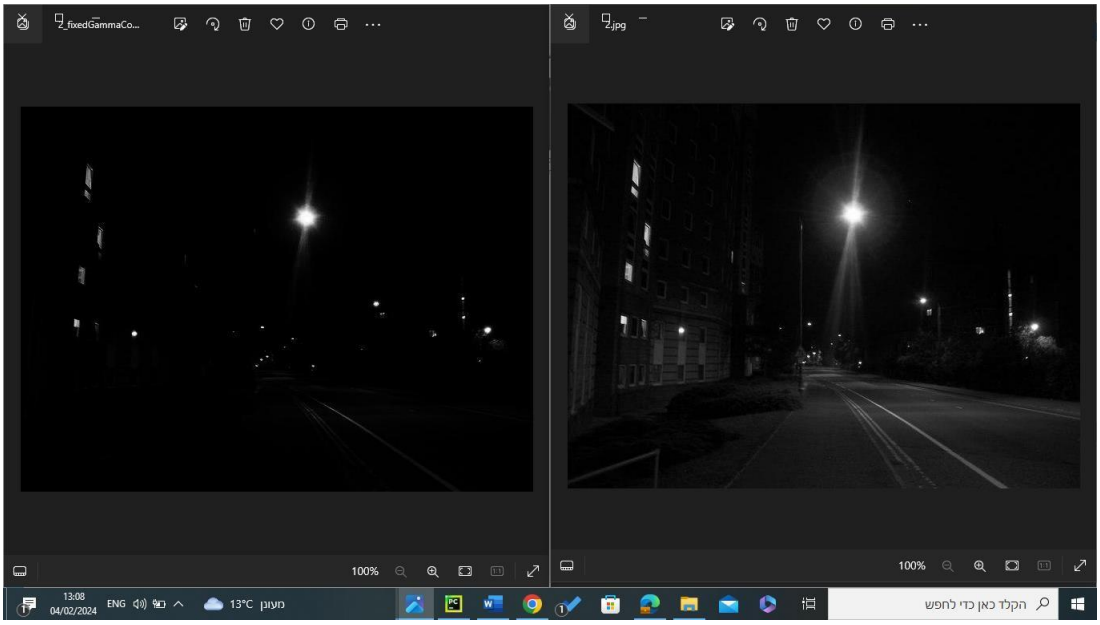


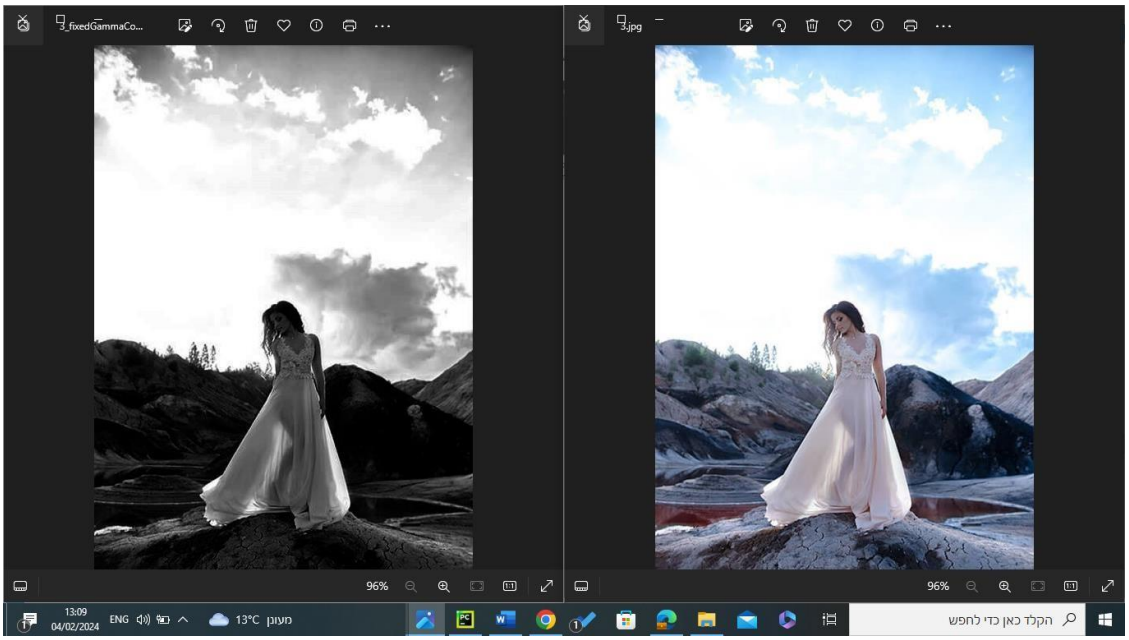


Gamma = 2.5:

In here we can see the images getting darker a lot, thus its not good for images with dark backgrounds and a lot of dark pixels, however with bright images it seems it gives a little more definition


1_fixedGammaCo...
1.png
100%
100%
13:07
04/02/2024
ENG
ILS/EUR +0.92%
הקלד כאן כדי לחפש


2_fixedGammaCo...
2.jpg
100%
100%
13:08
04/02/2024
ENG
13°C מעונן
הקלד כאן כדי לחפש



A) And so I have chosen that these are the corrections to be made that give the best results in my own eyes:

Image 1: Histogram Equalization. I chose this because it has the most definition out of all the images and we can see the most details

Image 2: Brightness and Contrast with alpha = 1.5 and beta = 50.

I chose this because this shows us the clearest image possible without adding too much noise , I noticed if we do gamma correction with 0.5 we may get a clearer view of the building however we add more background noise to the black background.

Image 3: Gamma with value of 2.5 (it gives it just a tiny bit more definition which is in my opinion an enhancement).

Question 2 :

a) *I picked 3 matching point for the puzzle with affine transformation*
And 4 for the puzzle with homography picking the best point by trying to
Pick the corner points and critical point that I need to align the two pictures with.

b) *Called the function prepare puzzle to prepare our data which represented by*
The matching point, is affine or not and the number of images:
matches, is_affine, n_images = prepare_puzzle(puzzle)

c) *This function gets whether the puzzle from affine type or not and the matching point*
For the I image in the puzzle, and we will get the transform matrix using cv2 will give it
Source and destination point, and it will calculate our transformation matrix using that
cv2.getAffineTransform(src_points, dst_points) if it is an affine puzzle
cv2.getPerspectiveTransform(src_points, dst_points) else

d) *inverse_transform_target_image in this function we will use the matrix extracted form the previous get_transform and apply in the image I used the :*
cv2.warpAffine(target_img, original_transform, output_size, flags=cv2.WARP_INVERSE_MAP) to affine transformation
*which takes the target image we want to apply on it the transformation matrix in this case we get it from get_transform and if it's affine transformation it's 2x3 matrix ,**output_size** the shape of the final puzzle so we can use the shape of image 1, and finally the flags this flag tell to do inverse to the transformation matrix before applying the*
transformation as we learned we need to do it because when mapping the points to the final stitched image going to be aligned.

e) *to implement the stich function we will first need to define a mask by doing this :*
mask = np.any(img2 != 0, axis=-1)
this will go through each pixel in the image 2 and if it not black (!=0) we will get in this pixel a true value else will get a false value, and we will use this as a condition to our np.where function :
np.where(mask, img2, img1) =(mask=condition)
this function will iterate through each pixel and take the pixel of the image 2 if the condition is true (the mask in this pixel is true) else will take the pixel from the first image by doing that we will ensure that just the transformed image that we need to stich to the final result will get stitched and not the black pixels around the image 2 also and that going to prevent covering the image one which going to be under it.