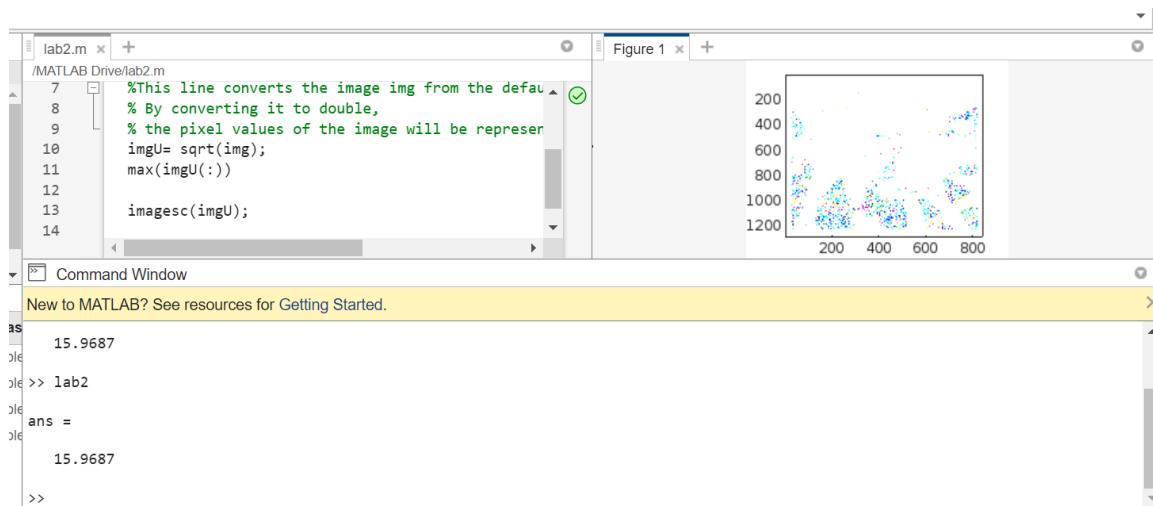


Line 2) The `imread` function is used to read the image, and it returns two outputs: `img` and `map`. `img` represents the image data, while `map` contains the colormap if the image is indexed.

Line 3) This line converts the image `img` from the default data type to double precision. By converting it to double, the pixel values of the image will be represented as floating-point numbers.

Line 4) This line calculates the square root of each pixel value in the `img` image. The `sqrt` function is applied element-wise to the image matrix, resulting in a new image `imgU` where each pixel value is the square root of the corresponding pixel value in the original image.

Line 5) This line finds the maximum value in the `imgU` image. The `(:)` syntax is used to linearize the image into a column vector, allowing the `max` function to find the maximum value across all the pixels in the image.

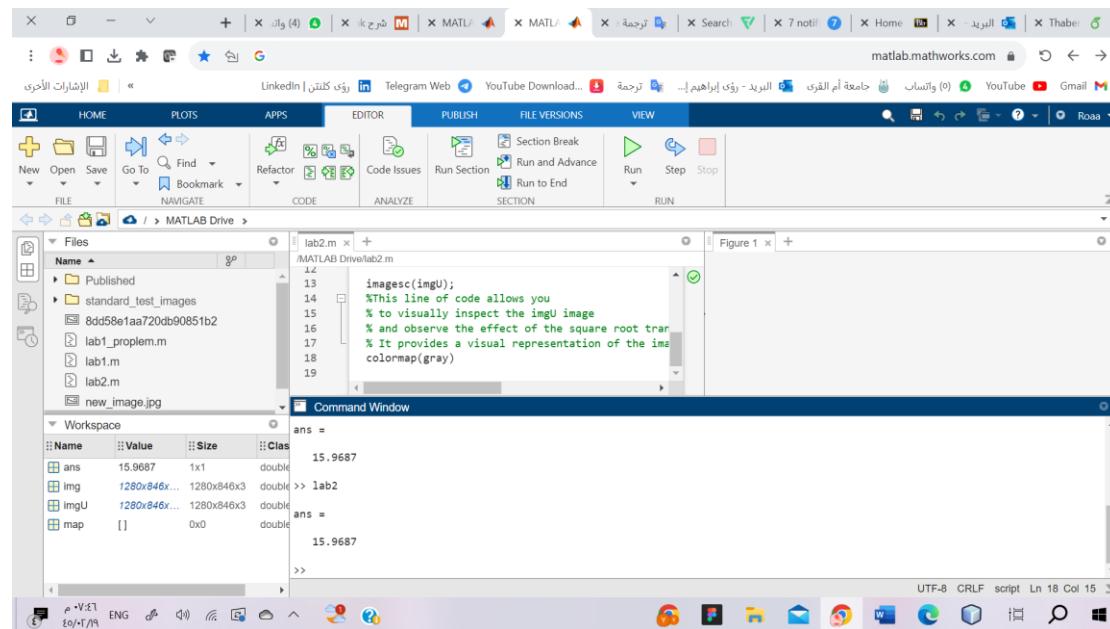


imagesc: imagesc is a MATLAB function used to display an image with scaled colors. It is commonly used for visualizing images with a color scale that represents the intensity values of the image.

imgU: imgU is the input image that is being displayed using imagesc. It represents the image where the square root of each pixel value has been calculated, as mentioned in the code you provided earlier.

When the line imagesc(imgU); is executed, it generates a figure window displaying the image imgU. The color scale of the image will be automatically adjusted to represent the range of intensity values in imgU, with lower values shown as darker shades and higher values shown as brighter shades.

This line of code allows you to visually inspect the imgU image and observe the effect of the square root transformation on the pixel intensities. It provides a visual representation of the image, making it easier to analyze and interpret the data.

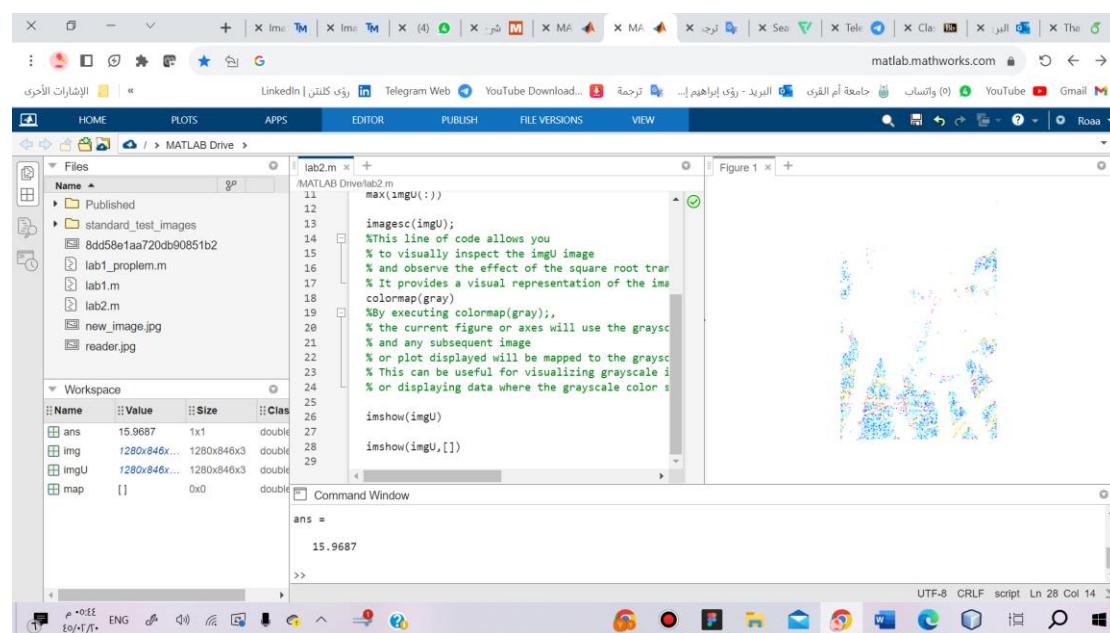
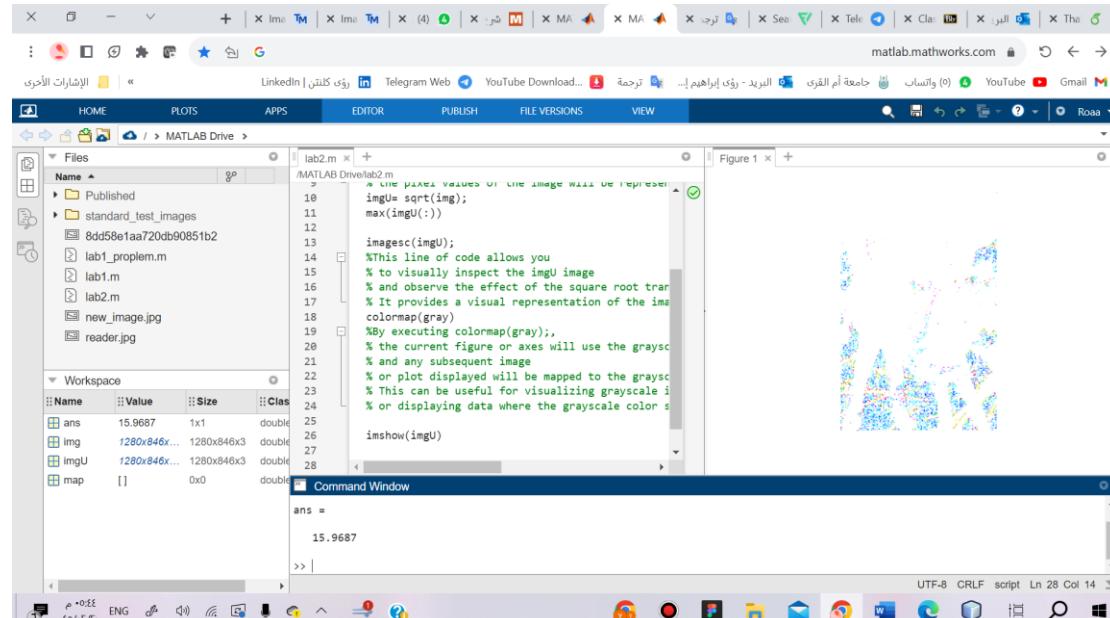


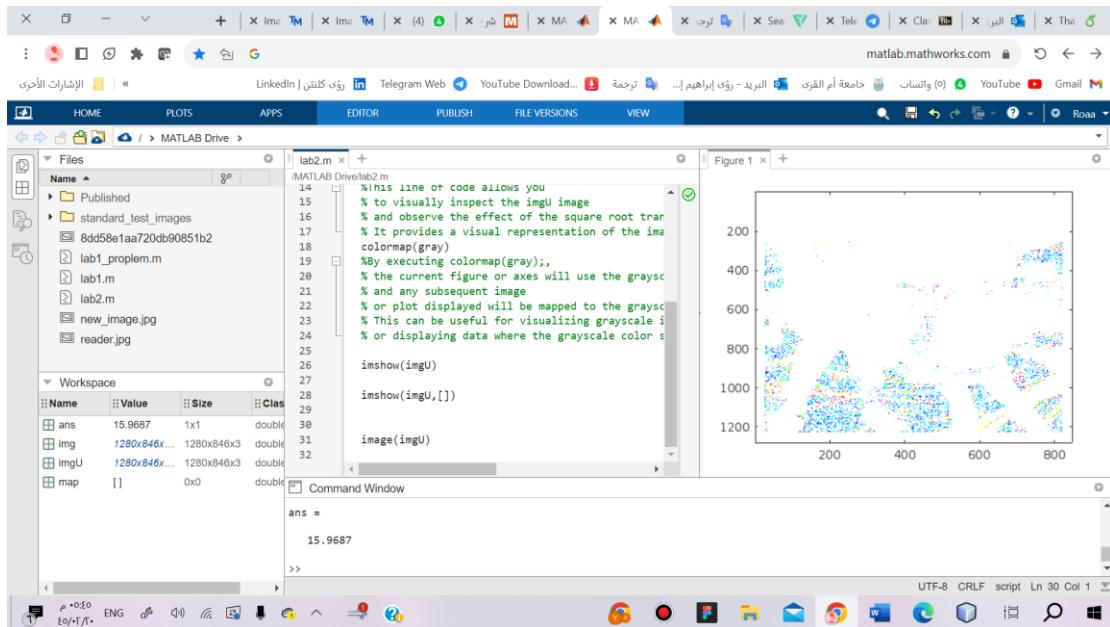
The code colormap(gray); is used in MATLAB to set the colormap of the current figure or axes to grayscale.

In MATLAB, a colormap is a set of colors that are used to represent the intensity values of an image or data. By default, MATLAB uses the "jet" colormap, which represents lower values with blue tones, intermediate values with green and yellow, and higher values with red tones.

However, in the code `colormap(gray);`, the gray colormap is specified. The "gray" colormap consists of shades of gray ranging from black for lower values to white for higher values. This colormap is often used to display grayscale images or to represent data where the intensity values are best represented by a grayscale color scheme.

By executing `colormap(gray);`, the current figure or axes will use the grayscale colormap, and any subsequent image or plot displayed will be mapped to the grayscale color scale. This can be useful for visualizing grayscale images or displaying data where the grayscale color scheme is appropriate.





`imshow(imgU)`: This line uses the `imshow` function to display the image `imgU`. The `imshow` function automatically scales the intensity values of the image to fit the display range. It adjusts the colormap and color scaling based on the minimum and maximum values in `imgU`. By default, it uses the "truecolor" display mode for RGB images or the grayscale display mode for grayscale images.

`imshow(imgU, [])`: This line also uses the `imshow` function, but with an additional argument `[]`. The `[]` argument is used to specify the display range. When `[]` is passed, MATLAB automatically scales the intensity values of `imgU` to the full display range. In other words, it adjusts the colormap and color scaling based on the minimum and maximum values in `imgU` to utilize the full dynamic range of the colormap.

`image(imgU)`: This line uses the `image` function to display the image `imgU`. The `image` function is similar to `imshow` and can be used to display images. However, unlike `imshow`, the `image` function does not automatically adjust the colormap or color scaling. You would need to manually set the colormap and color scaling using the `colormap` function or by specifying additional arguments to the `image` function.

رئي إبراهيم كلتن لاب ٢  
معالجة صور شعبة ١ نظري ، عملي شعبة ==> شعبة ١

The screenshot shows the MATLAB IDE interface. The code editor displays a script named lab2.m. The command window shows the execution of the script, displaying variable assignments and their sizes. The workspace browser shows various variables and their values.

```
>> lab2
ans =
15.9687
Name      Size            Bytes  Class
ans        1x1              8    double
img       1280x846x3     25989120 double
img1      350x250x3      262500  uint8
img2      677x480x3      974880  uint8
imgU     1280x846x3     25989120 double
map       []                0    double

>> lab2
ans =
15.9687
Name      Size            Bytes  Class
ans        1x1              8    double
img       1280x846x3     25989120 double
img1      350x250x3      262500  uint8
img2      677x480x3      974880  uint8
imgU     1280x846x3     25989120 double
map       []                0    double
```

The screenshot shows the MATLAB IDE interface. The code editor displays the same script lab2.m. The command window shows the execution of the script, but at line 57, it displays an error message: "File: lab2.m Line: 57 Column: 25 Invalid expression. Check for missing multiplication operator, missing or unbalanced delimiters, or other syntax error. To construct matrices, use brackets instead of parentheses." The workspace browser shows variables and their values.

```
>> lab2
File: lab2.m Line: 57 Column: 25
Invalid expression. Check for missing multiplication
operator, missing or unbalanced delimiters, or other syntax
error. To construct matrices, use brackets instead of
parentheses.

>> lab2
ans =
15.9687
Name      Size            Bytes  Class
ans        1x1              8    double
img       1280x846x3     25989120 double
img1      350x250x3      262500  uint8
img2      677x480x3      974880  uint8
imgU     1280x846x3     25989120 double
map       []                0    double
```

## رؤى إبراهيم كلتن لاب ٢ معالجة صور شعبة ١ نظري ، عملي شعبة ==> شعبة ١

The screenshot shows the MATLAB IDE interface. The left pane displays the 'lab2.m' script with code related to image processing. The right pane shows the 'Command Window' with the following output:

```

ans = 1x1 8 double
img 1280x846x3 25989120 double
img1 350x250x3 262500 uint8
img2 677x480x3 974880 uint8
imgU 1280x846x3 25989120 double
map 0x0 0 double

>> lab2
ans =
15.9687

Name Size Bytes Class Attributes
ans 1x1 8 double
img 1280x846x3 25989120 double
img1 350x250x3 262500 uint8
img2 677x480x3 974880 uint8
imgU 1280x846x3 25989120 double
map 0x0 0 double

ans =
1280 846 3

```

The screenshot shows the MATLAB IDE interface. The left pane displays the 'lab2.m' script. The right pane shows the 'Command Window' with the following output:

```

>> lab2
ans =
15.9687

Name Size Bytes Class Attributes
ans 1x1 8 double
img 1280x846x3 25989120 double
img1 350x250x3 262500 uint8
img2 677x480x3 974880 uint8
imgU 1280x846x3 25989120 double
map 0x0 0 double

ans =
1280 846 3

```

Below the command window, an error message is displayed:

```

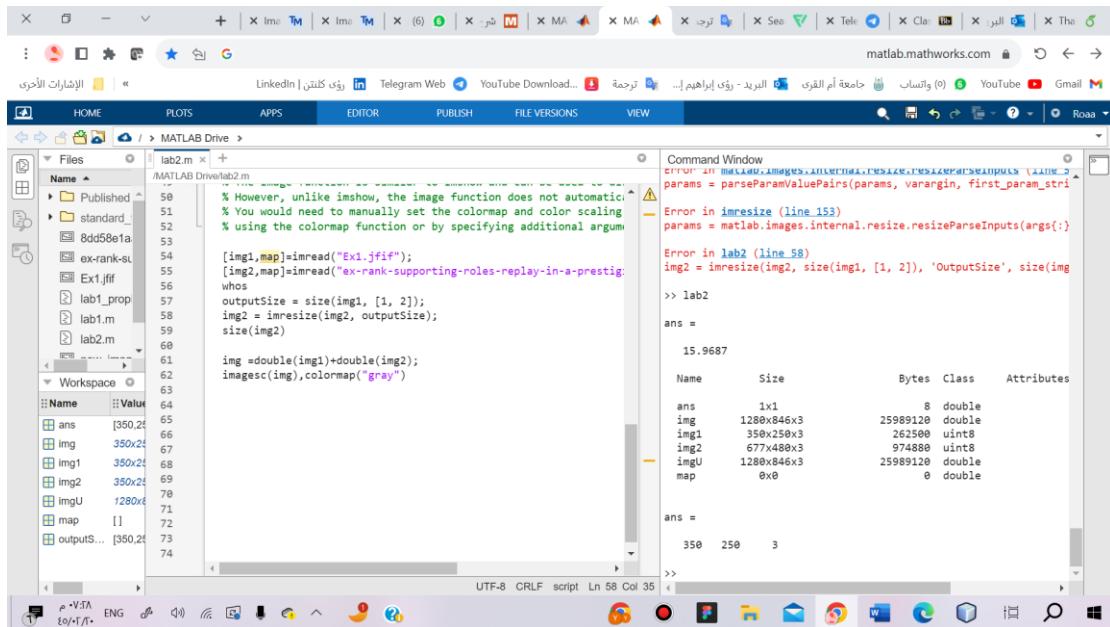
Arrays have incompatible sizes for this operation.

Error in lab2 (line 68)
img =double(img1)+double(img2);

Related documentation

```

رؤى إبراهيم كلنتن لاب ٢  
معالجة صور شعبة ١ نظري ، عملي شعبة ٤ ==> شعبة ١



Certainly! Let's go through the code step by step to explain what each part does:

```
[img1, map] = imread("Ex1.jfif"); and [img2, map] = imread("ex-rank-supporting-roles-replay-in-a-prestigious-school.jpg");
```

These lines read the images from the files "Ex1.jfif" and "ex-rank-supporting-roles-replay-in-a-prestigious-school.jpg", respectively, using the `imread` function.

The resulting image data is stored in `img1` and `img2`, and the colormap (if present) is stored in `map`.

`whos`:

This command displays information about the variables currently in the workspace.

It shows details such as the name, size, bytes, and class of each variable.

In this case, it provides information about `img1`, `img2`, `map`, and other variables.

```
outputSize = size(img1, [1, 2]);
```

The `size` function is used to determine the dimensions of `img1`.

The `[1, 2]` argument specifies that we want to retrieve the size in the first and second dimensions (height and width) of `img1`.

The resulting size is assigned to the `outputSize` variable.

```
img2 = imresize(img2, outputSize);
```

The imresize function is used to resize img2 to match the dimensions of img1.

The outputSize variable, which contains the desired dimensions, is passed as the second argument to imresize.

This ensures that img2 is resized to have the same size as img1.

size(img2):

This line displays the size of img2 after the resizing operation.

It shows the dimensions (height, width, and color channels) of the resized img2.

img = double(img1) + double(img2);

The double function is used to convert img1 and img2 from the original data type (likely uint8) to double precision.

This conversion allows for performing element-wise addition between the two images.

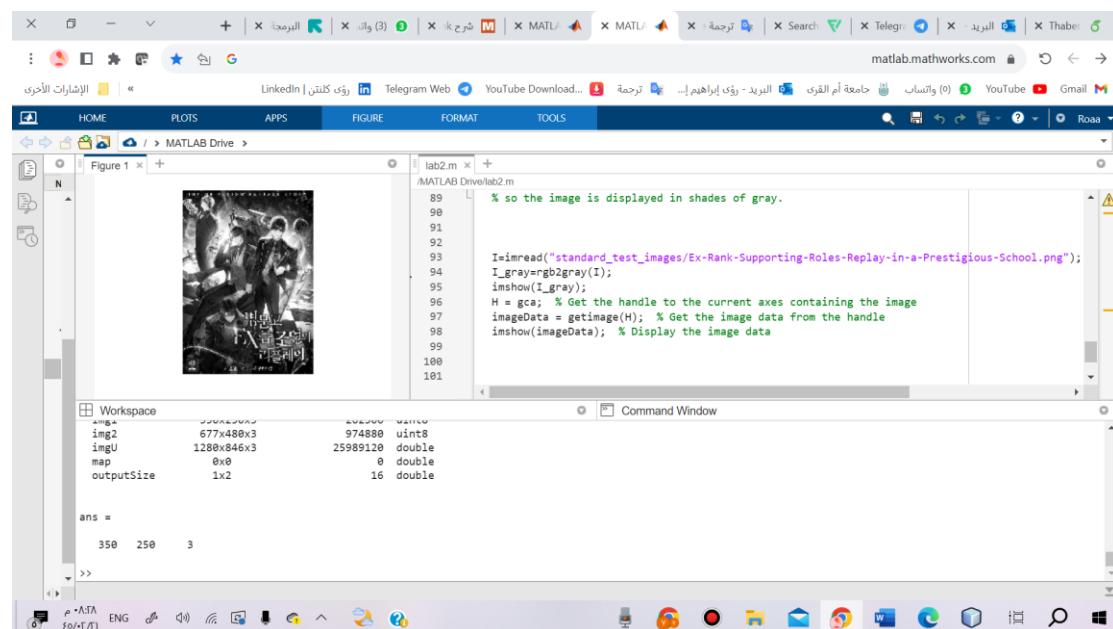
The resulting sum is stored in the img variable.

imagesc(img), colormap("gray"):

The imagesc function is used to display the img variable as an image.

The colormap("gray") command sets the colormap to grayscale, so the image is displayed in shades of gray.

Overall, the code reads two images, resizes one of them to match the dimensions of the other, adds the two images together, and displays the resulting image. The goal is to visualize the combined effect of adding the pixel values of the two original images.



The screenshot shows the MATLAB desktop environment. The top menu bar includes HOME, PLOTS, APPS, FIGURE, FORMAT, and TOOLS. The central workspace shows a figure window titled 'Figure 1' displaying a grayscale image of a school scene. To the right of the figure is a code editor window with the file 'lab2.m'. The code reads an image, converts it to grayscale, and then displays it using the imagesc function. Below the code editor is the Command Window, which shows the workspace variables and their sizes. The bottom of the screen shows the Windows taskbar with various icons.

```
% so the image is displayed in shades of gray.  
I=imread("standard_test_images/Ex-Rank-Supporting-Roles-Replay-in-a-Prestigious-School.png");  
I_gray=rgb2gray(I);  
imshow(I_gray);  
H = gca; % Get the handle to the current axes containing the image  
imageData = getimage(H); % Get the image data from the handle  
imshow(imageData); % Display the image data
```

Name	Type	Size	Value
img	uint8	677x488x3	974880
img2	uint8	1280x846x3	25989120
map	double	0x0	0
outputSize	double	1x2	16
ans	double	350 250 3	

```
H = gca;:
```

The `gca` function stands for "get current axes" and returns the handle to the current axes object that contains the image.

The handle is stored in the variable `H` for further use.

```
imageData = getimage(H);:
```

The `getimage` function is used to retrieve the image data from the axes handle `H`.

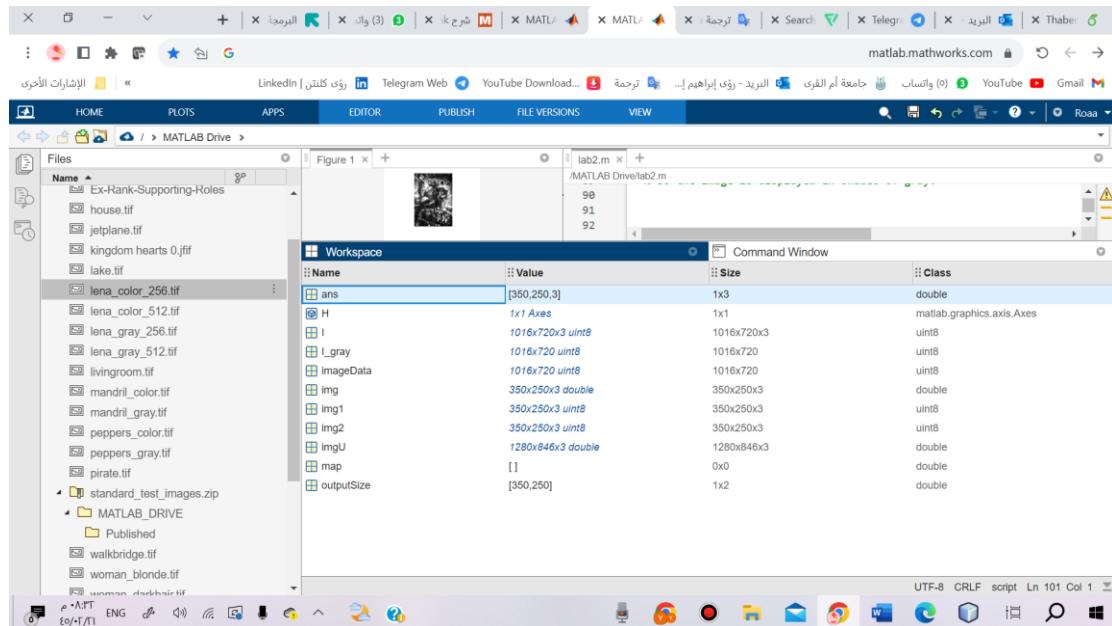
It extracts the image data displayed in the axes object, which in this case is the grayscale image `I_gray`.

The retrieved image data is stored in the variable `imageData`.

```
imshow(imageData);:
```

The `imshow` function is called again, but this time it displays the image data stored in `imageData`.

Since `imageData` contains the same grayscale image as before, this line essentially re-displays the image.



رؤى إبراهيم كلتن لاب ٢  
معالجة صور شعبة ١ نظري ، عملي شعبة ==> شعبة ١

The screenshot shows the MATLAB IDE interface. The Command Window displays the following code and variable information:

```

ans =
15.9687
Name      Size
H          1x1
H1         256x1
H_Equ     256x256
I          1016x720x3
I1         256x256x3
I1_gray   256x256
I_gray    1016x720
ans       1x1
imageData 1016x720
img        1280x846x3
img1      350x250x3
img2      677x480x3
imgU     1280x846x3
map        0x0
outputSize 1x2

```

The Workspace browser shows the following variables:

Name	Value	Size	Class
ans	[350,250,3]	1x3	double
H	1x1 Axes	1x1	matlab.graphics.axis.Axes
H1	256x1 double	256x1	double
H_Equ	256x256 uint8	256x256	uint8
I	1016x720x3 uint8	1016x720x3	uint8

The Editor tab contains the script file `lab2.m` with the following code:

```

% MATLAB Drive\lab2.m
% I=rgb2gray(I);
95 imshow(I_gray);
96 H = gca; % Get the handle to the current axes containing the image
97 imageData = getimage(H); % Get the image data from the handle
98 imshow(imageData); % Display the image data
99
100 I1imread('/MATLAB Drive/standard_test_images/lena_color_256.tif');
101
102 I1_gray=rgb2gray(I1);
103 imshow(I1_gray);
104
105
106
107
108
109

```

The screenshot shows the MATLAB IDE interface. The Command Window displays the same code and variable information as the previous screenshot.

The Workspace browser shows the following variables:

Name	Value	Size	Class
ans	[350,250,3]	1x3	double
H	1x1 Axes	1x1	matlab.gr...
H1	256x1 double	256x1	double
H_Equ	256x256 ul...	256x256	uint8
I	1016x720x3...	1016x720x3	uint8
I1	256x256x3...	256x256x3	uint8
I1_gray	256x256 ul...	256x256	uint8
I_gray	1016x720 u...	1016x720	uint8
imageD...	1016x720 u...	1016x720	uint8
img	350x250x3...	350x250x3	double
img1	350x250x3...	350x250x3	uint8
img2	350x250x3...	350x250x3	uint8
imgU	1280x846x3...	1280x846x3	double
map	[]	0x0	double
outputS...	[350,250]	1x2	double

The Editor tab contains the script file `lab2.m` with the following code:

```

ans =
15.9687
Name      Size
H          1x1
H1         256x1
H_Equ     256x256
I          1016x720x3
I1         256x256x3
I1_gray   256x256
I_gray    1016x720
ans       1x1
imageData 1016x720
img        1280x846x3
img1      350x250x3
img2      677x480x3
imgU     1280x846x3
map        0x0
outputSize 1x2

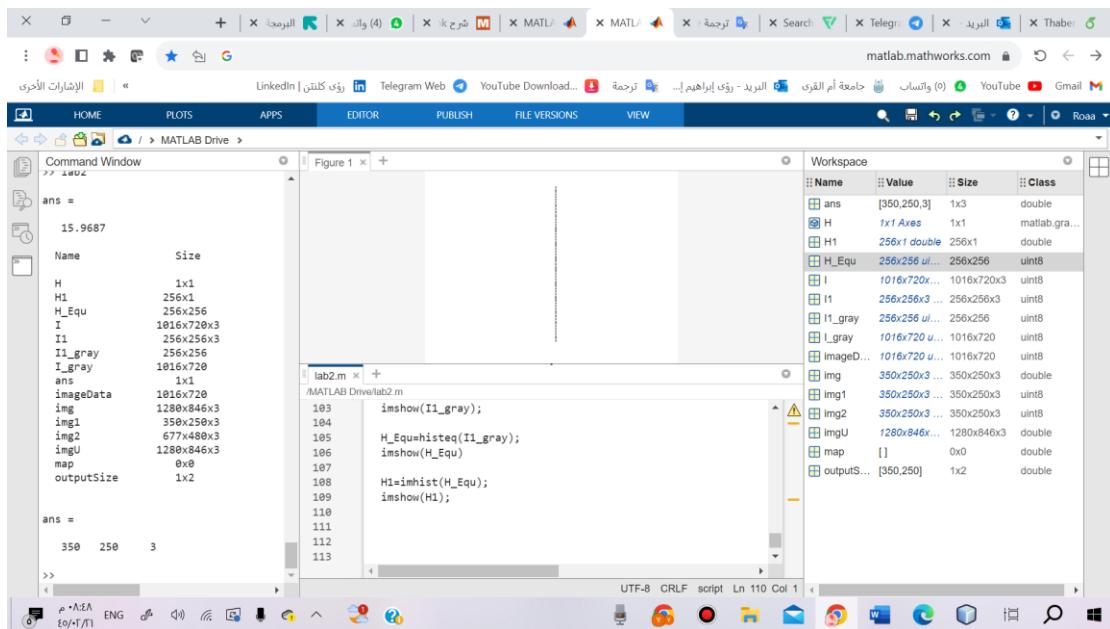
```

```

101
102 I1_gray=rgb2gray(I1);
103 imshow(I1_gray);
104
105 H_Equ=histeq(I1_gray);
106 imshow(H_Equ)
107
108
109

```

رؤى إبراهيم كلنتن لاب ٢  
معالجة صور شعبة ١ نظري ، عملي شعبة ==> شعبة ١



`H_Equ = histeq(I1_gray);;`

The `histeq` function performs histogram equalization on the grayscale image `I1_gray`.

Histogram equalization enhances the contrast of the image by redistributing pixel intensities.

The resulting equalized image is stored in the variable `H_Equ`.

`imshow(H_Equ);;`

The `imshow` function is called to display the equalized image `H_Equ`.

This line shows the equalized image in a new figure window.

`H1 = imhist(H_Equ);;`

The `imhist` function is used to compute the histogram of the equalized image `H_Equ`.

The histogram represents the frequency distribution of pixel intensities in the image.

The resulting histogram data is stored in the variable `H1`.

`imshow(H1);`

The `imshow` function is called to display the histogram data `H1`.

Since `H1` contains the histogram data, this line shows the histogram plot in a figure window.

#####

```
%lab2  
[img, map]=imread("reader.jpg");
```

```
%The imread function is used to read the image,  
% and it returns two outputs: img and map. img represents the image data,  
% while map contains the colormap if the image is indexed.  
img= double(img);  
%This line converts the image img from the default data type to double  
precision.  
% By converting it to double,  
% the pixel values of the image will be represented as floating-point  
numbers.  
imgU= sqrt(img);  
%This line calculates the square root  
% of each pixel value in the img image.  
% The sqrt function is applied element-wise to the image matrix,  
% resulting in a new image imgU where each pixel value is the square root  
of the corresponding pixel value in the original image.  
max(imgU(:))  
%This line finds the maximum value in the imgU image.  
% The () syntax is used to linearize the image into a column vector,  
% allowing the max function to find the maximum value across all the pixels  
in the image.  
imagesc(imgU);  
%This line of code allows you  
% to visually inspect the imgU image  
% and observe the effect of the square root transformation on the pixel  
intensities.  
% It provides a visual representation of the image, making it easier to  
analyze and interpret the data.  
colormap(gray)  
%By executing colormap(gray);,  
% the current figure or axes will use the grayscale colormap,  
% and any subsequent image  
% or plot displayed will be mapped to the grayscale color scale.  
% This can be useful for visualizing grayscale images  
% or displaying data where the grayscale color scheme is appropriate.  
  
imshow(imgU)  
%This line uses the imshow function  
% to display the image imgU.  
% The imshow function automatically scales  
% the intensity values of the image  
% to fit the display range.  
% It adjusts the colormap and color scaling  
% based on the minimum and maximum values in imgU.  
% By default, it uses the "truecolor" display mode for RGB images  
% or the grayscale display mode for grayscale images.  
  
imshow(imgU,[])  
% This line also uses the imshow function,  
% but with an additional argument [].  
% The [] argument is used to specify the display range.  
% When [] is passed,  
% MATLAB automatically scales the intensity values of imgU to the full  
display range.  
% In other words,  
% it adjusts the colormap and color scaling  
% based on the minimum and maximum values in imgU to utilize the full  
dynamic range of the colormap.
```

```
image(imgU)
%This line uses the image function to display the image imgU.
% The image function is similar to imshow and can be used to display
images.
% However, unlike imshow, the image function does not automatically adjust
the colormap or color scaling.
% You would need to manually set the colormap and color scaling
% using the colormap function or by specifying additional arguments to the
image function.

[img1,map]=imread("Ex1.jfif");
[img2,map]=imread("ex-rank-supporting-roles-replay-in-a-prestigious-
school.jpg");
%read from file
%The resulting image data is stored in img1 and img2, and the colormap (if
present) is stored in map.
whos
%This command displays information about the variables currently in the
workspace.
%It shows details such as the name, size, bytes, and class of each
variable.
outputSize = size(img1, [1, 2]);
%The size function is used to determine the dimensions of img1.
%the [1, 2] argument specifies that we want to retrieve the size in the
first and second dimensions (height and width) of img1.
%The resulting size is assigned to the outputSize variable.

img2 = imresize(img2, outputSize);
%The imresize function is used to resize img2 to match the dimensions of
img1.
% The outputSize variable,
% which contains the desired dimensions,
% is passed as the second argument to imresize.
% This ensures that img2 is resized to have the same size as img1
size(img2)
%This line displays the size of img2 after the resizing operation.
% It shows the dimensions (height, width, and color channels) of the
resized img2.
img =double(img1)+double(img2);
%The double function is used to convert img1 and img2 from the original
data type (likely uint8) to double precision.
% This conversion allows for performing element-wise addition between the
two images.
% The resulting sum is stored in the img variable.
imagesc(img),colormap("gray")
%The imagesc function is used to display the img variable as an image.
% The colormap("gray") command sets the colormap to grayscale,
% so the image is displayed in shades of gray.
```

```
I=imread("standard_test_images/Ex-Rank-Supporting-Roles-Replay-in-a-
Prestigious-School.png");
I_gray=rgb2gray(I);
imshow(I_gray);
H = gca; % Get the handle to the current axes containing the image
imageData = getimage(H); % Get the image data from the handle
imshow(imageData); % Display the image data
```

```
I1=imread(' /MATLAB Drive/standard_test_images/lena_color_256.tif');

I1_gray=rgb2gray(I1);
imshow(I1_gray);

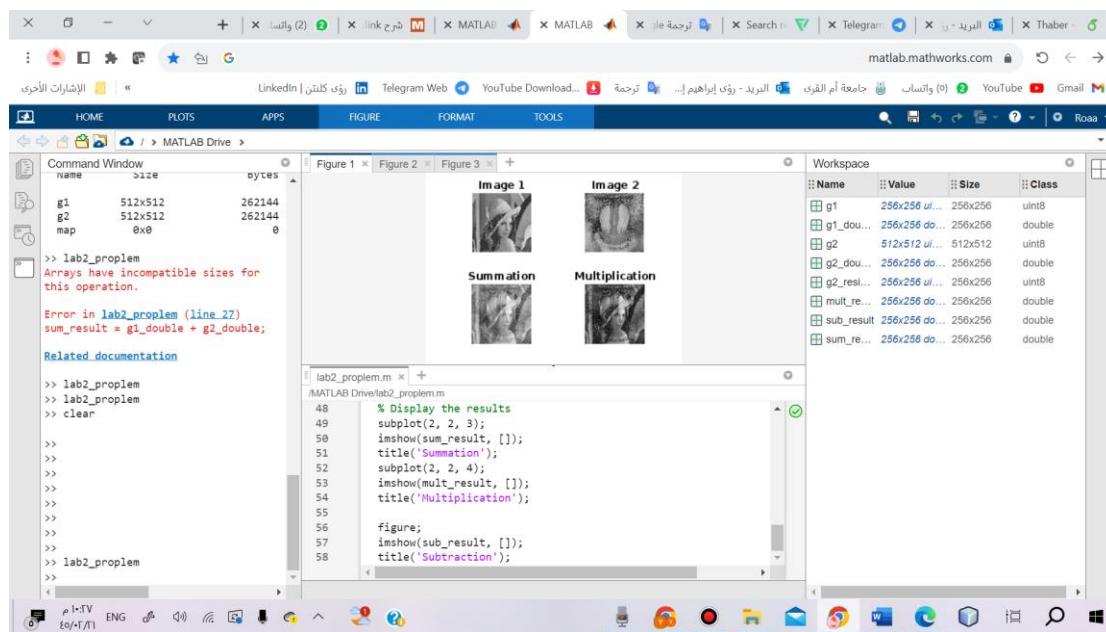
H_Equ=histeq(I1_gray);
%The histeq function performs histogram equalization
% on the grayscale image I1_gray.
% Histogram equalization enhances the contrast
% of the image by redistributing pixel intensities.
imshow(H_Equ)

H1=imhist(H_Equ);
%The imhist function is used to compute the histogram
% of the equalized image H_Equ.
% The histogram represents the frequency distribution
% of pixel intensities in the image.
imshow(H1);
```

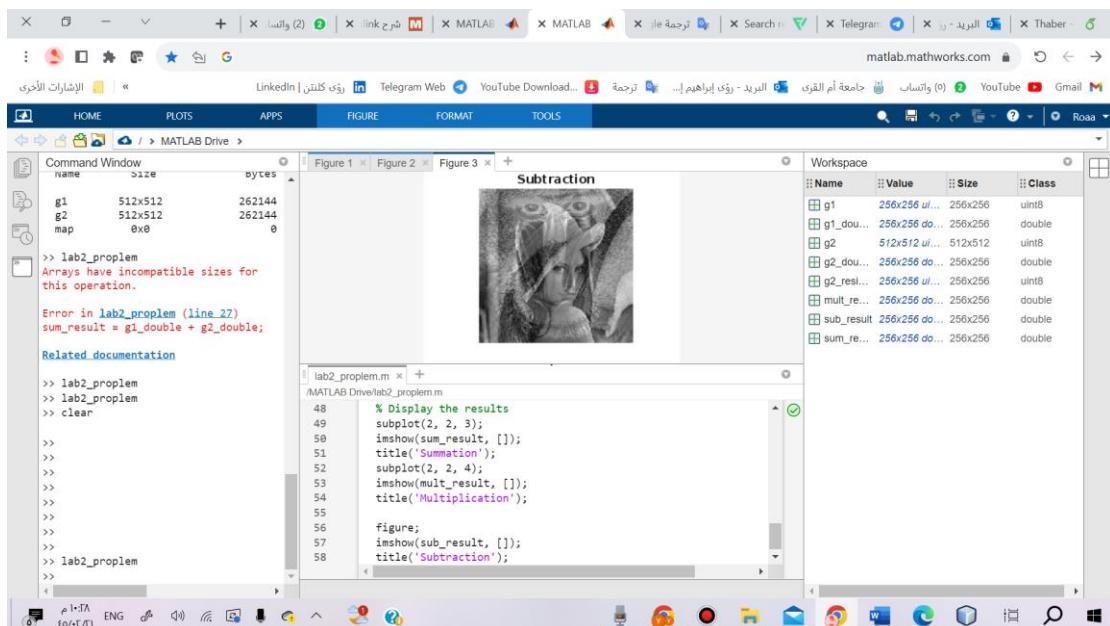
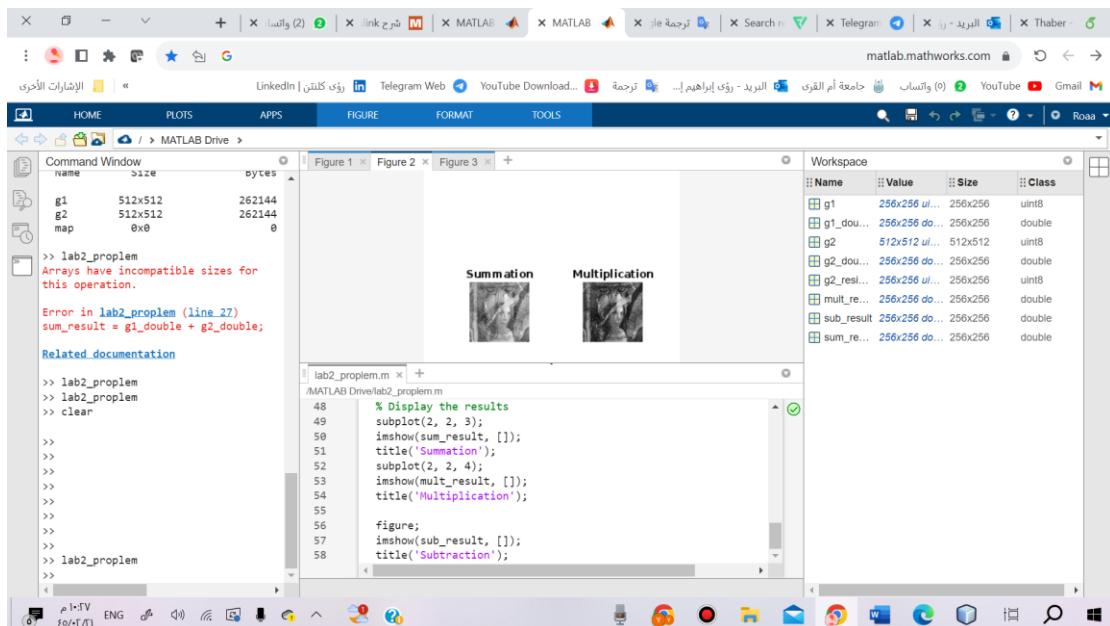
#### Problems:

- 1- Read two grayscale images with different sizes. Try several binary operations on them: summation, multiplication, subtraction. Display the results and explain them.
- 2- Read grayscale image them display the normal histogram for it

S1:



رؤى إبراهيم كلتن لاب ٢  
معالجة صور شعبة ١ نظري ، عملي شعبة ==> شعبة ١



```

g1 = imread("/MATLAB Drive/standard_test_images/lena_gray_256.tif");
g2 = imread("/MATLAB Drive/standard_test_images/mandril_gray.tif");

% [g1, map] = imread("/MATLAB Drive/standard_test_images/lena_gray_256.tif");
% [g1, map] = imread("/MATLAB Drive/standard_test_images/mandril_gray.tif");

% whos
% g3 = double(g1) + double(g2);

% Display the original images
figure;
subplot(2, 2, 1);
imshow(g1);
title('Image 1');
subplot(2, 2, 2);
imshow(g2);
title('Image 2');

```

```
% Resize g2 to match the size of g1
g2_resized = imresize(g2, size(g1));

% Convert the images to double precision for arithmetic operations
g1_double = double(g1);
g2_double = double(g2_resized);

% Perform binary operations
sum_result = g1_double + g2_double;
mult_result = g1_double .* g2_double;
sub_result = g1_double - g2_double;

% Display the results until 8 or double
subplot(2, 2, 3);
imshow(sum_result, []);
title('Summation');
subplot(2, 2, 4);
imshow(mult_result, []);
title('Multiplication');

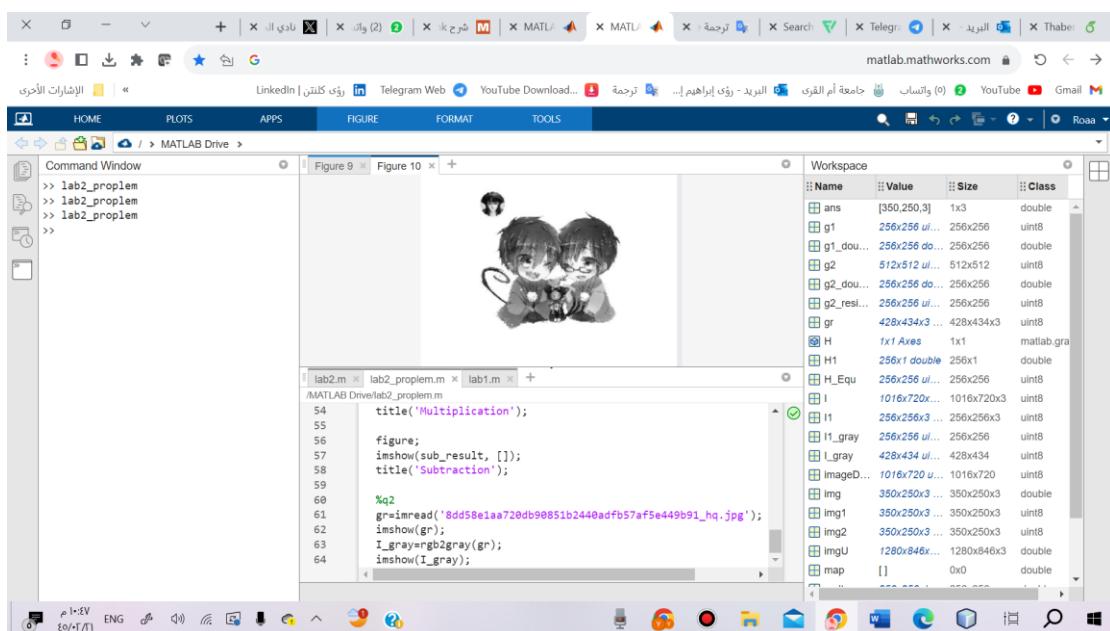
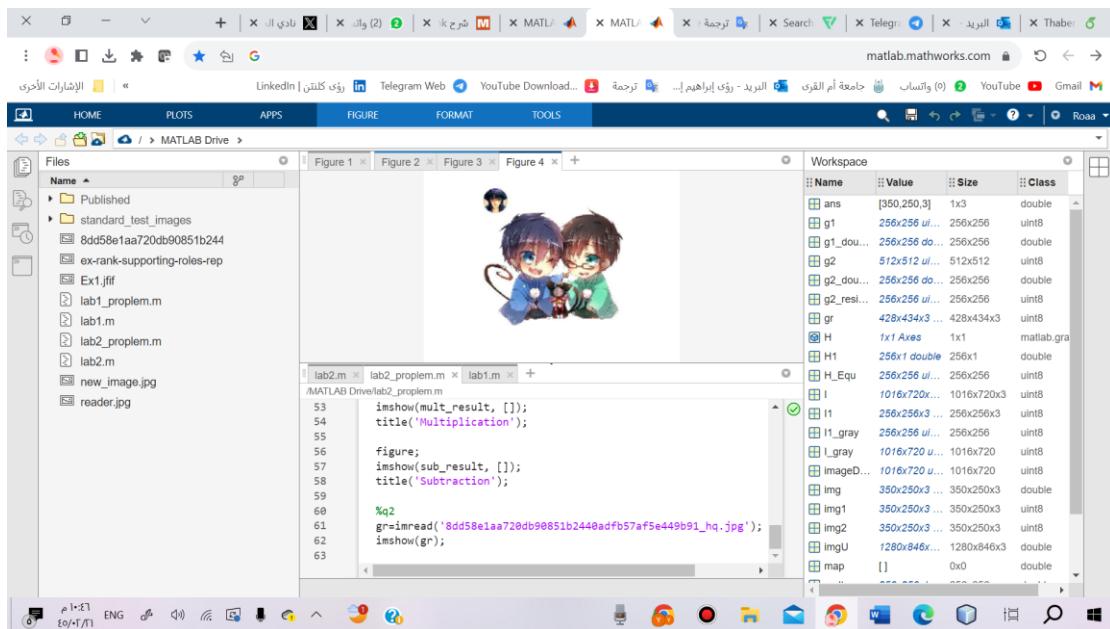
figure;
imshow(sub_result, []);
title('Subtraction');
% Perform binary operations
sum_result = g1_double + g2_double;
mult_result = g1_double .* g2_double;
sub_result = g1_double - g2_double;

% Display the results
subplot(2, 2, 3);
imshow(sum_result, []);
title('Summation');
subplot(2, 2, 4);
imshow(mult_result, []);
title('Multiplication');

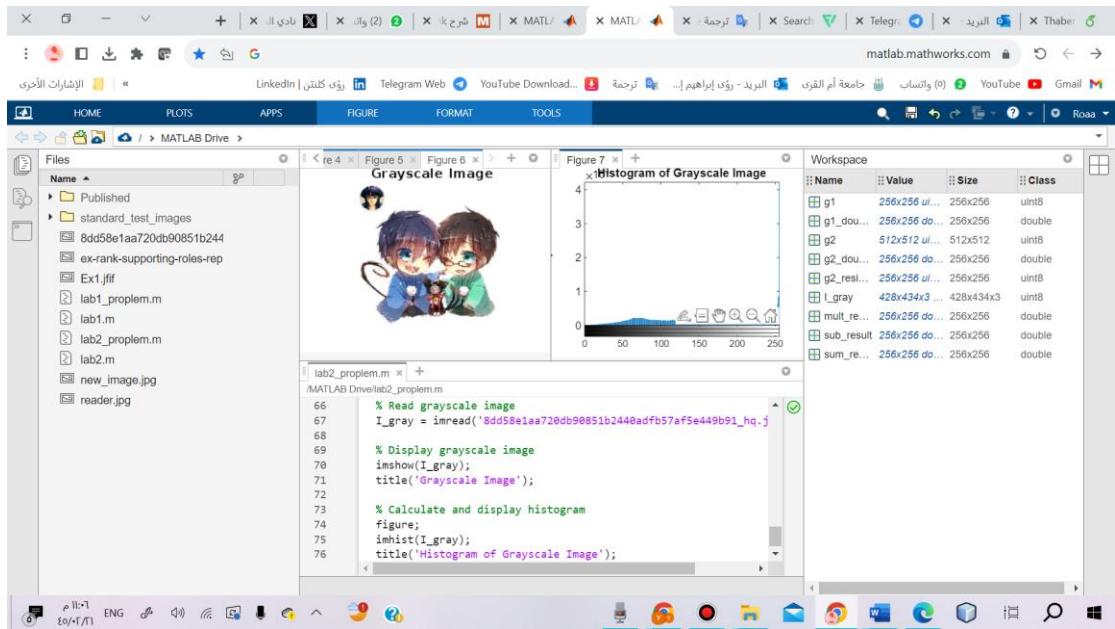
figure;
imshow(sub_result, []);
title('Subtraction');
```

S2:

رؤى إبراهيم كلتن لاب ٢  
معالجة صور شعبة ١ نظري ، عملي شعبة ==> شعبة ١



رؤى إبراهيم كلنتن لاب ٢  
معالجة صور شعبة ١ نظري ، عملي شعبة ==> شعبة ١



```
%q2
%gr=imread('8dd58e1aa720db90851b2440adfb57af5e449b91_hq.jpg');
%imshow(gr);
%I_gray=rgb2gray(gr);
%imshow(I_gray);

% Read grayscale image
I_gray = imread('8dd58e1aa720db90851b2440adfb57af5e449b91_hq.jpg');

% Display grayscale image
imshow(I_gray);
title('Grayscale Image');

% Calculate and display histogram
figure;
imhist(I_gray);
title('Histogram of Grayscale Image');
```