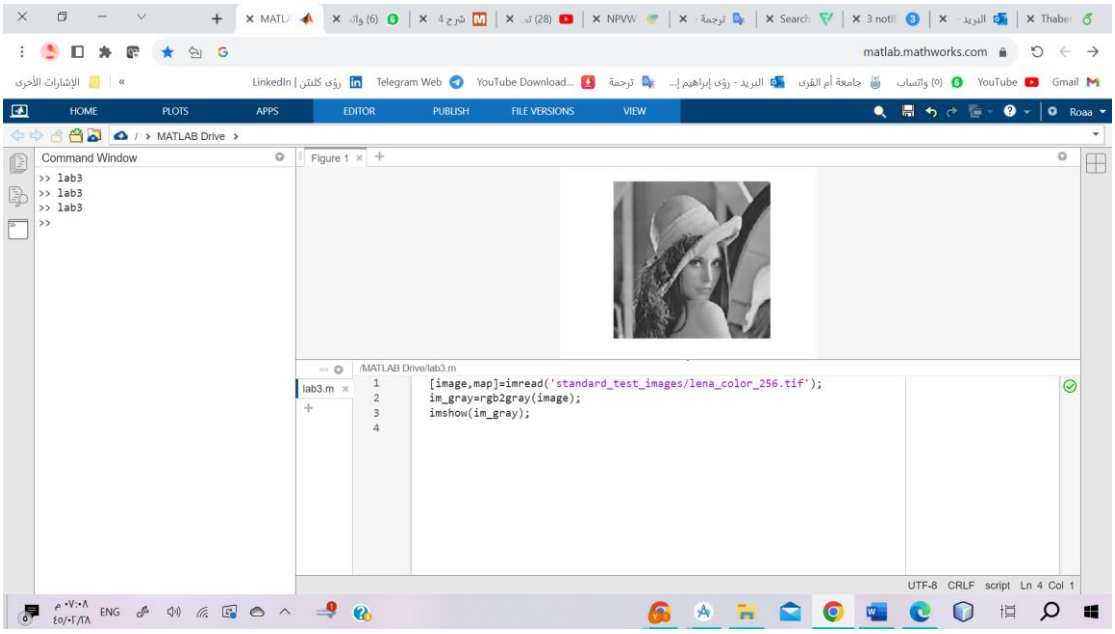
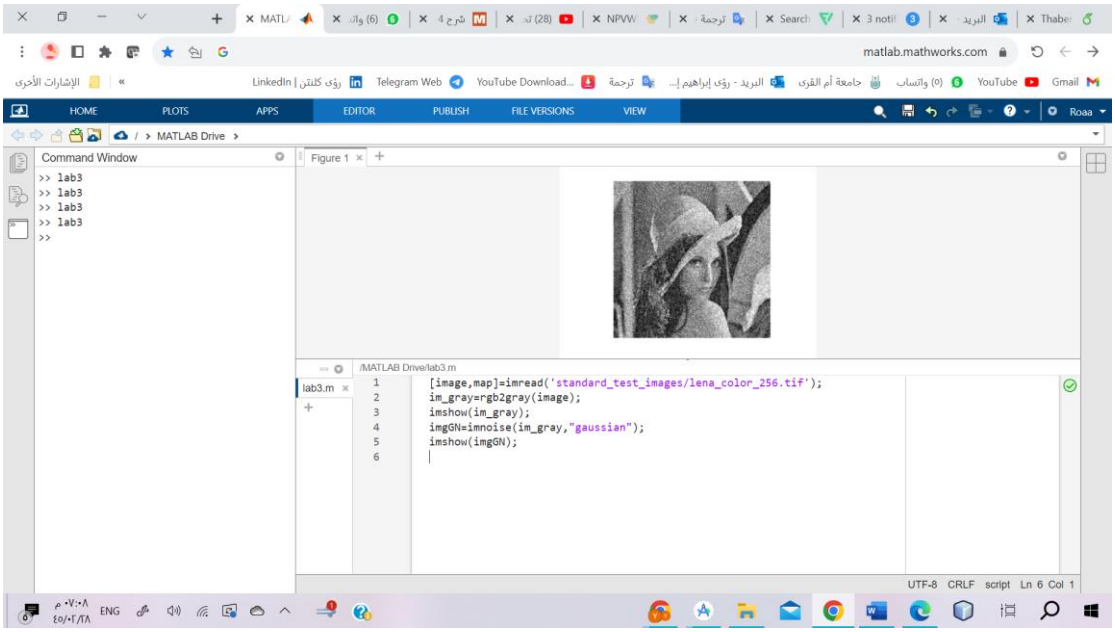
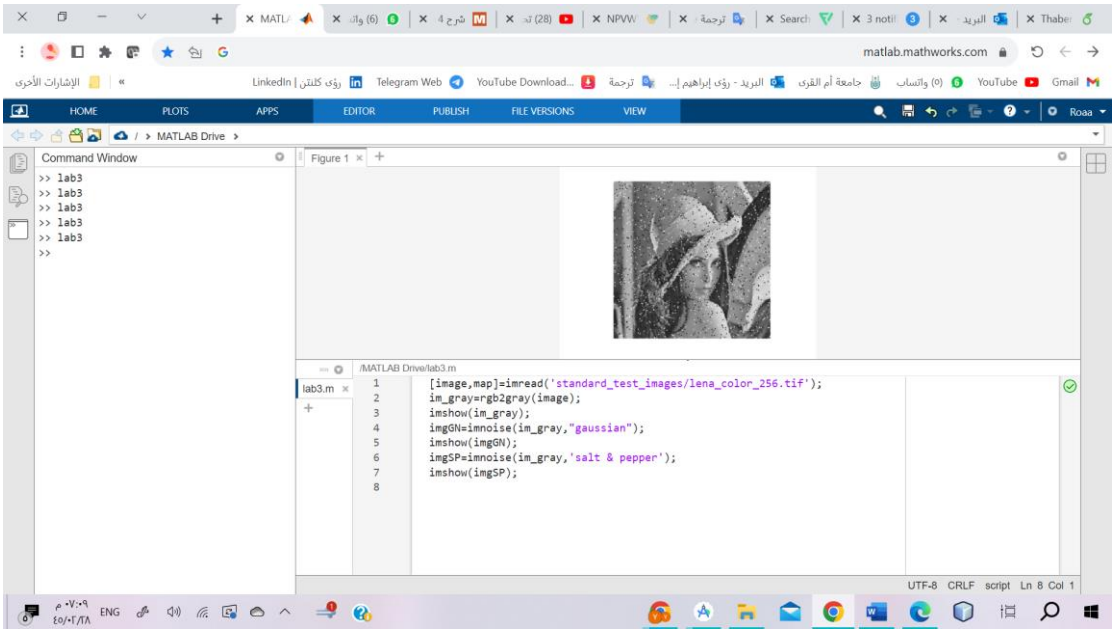
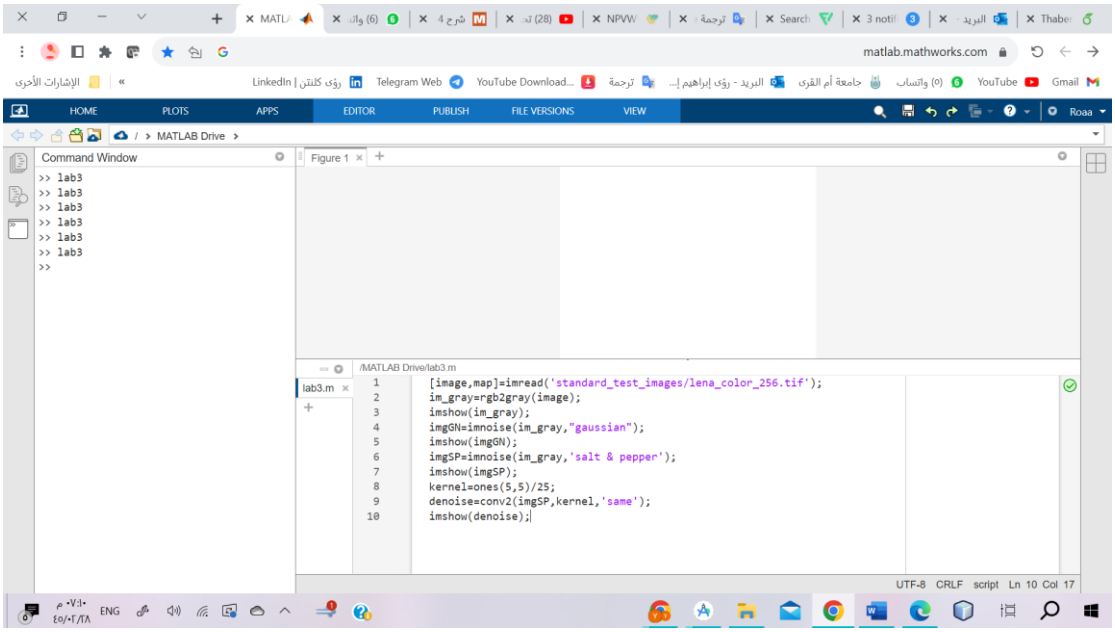
رؤى إبراهيم كلنتن 441004834
معالجة صور شعبة نظري 1 شعبة العملي 4

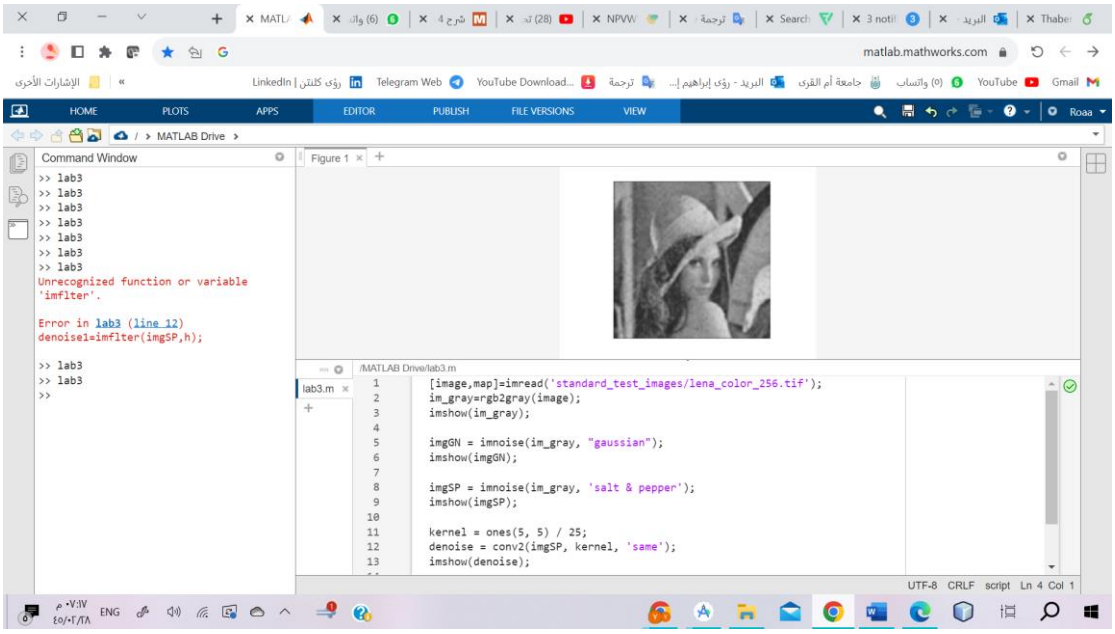


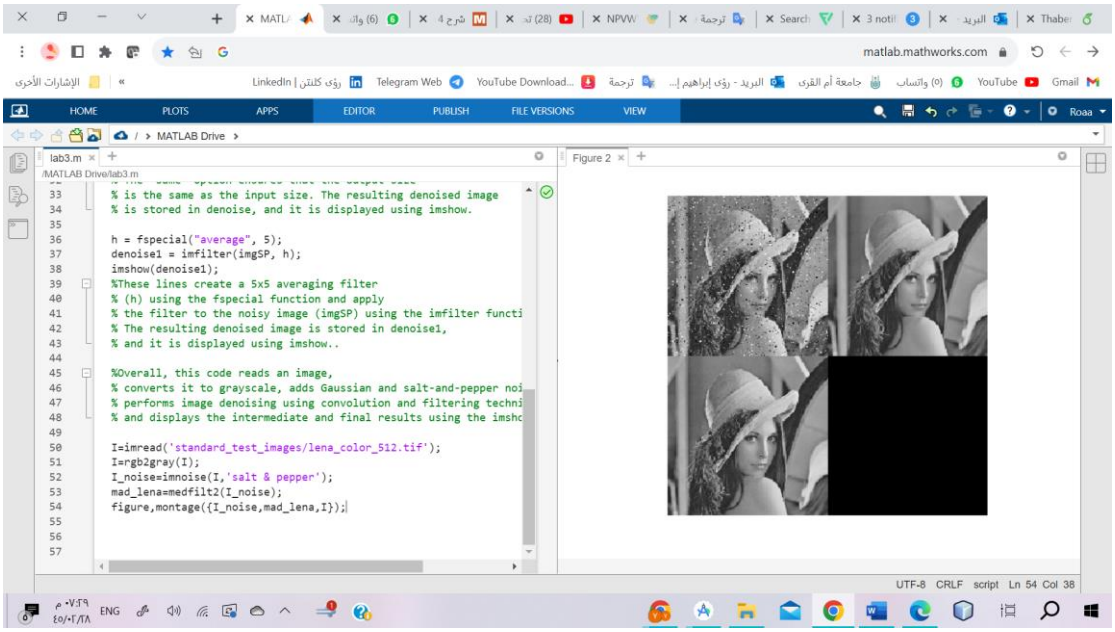


رؤى إبراهيم كلنتن 441004834
معالجة صور شعبة نظري 1 شعبة العملي 4







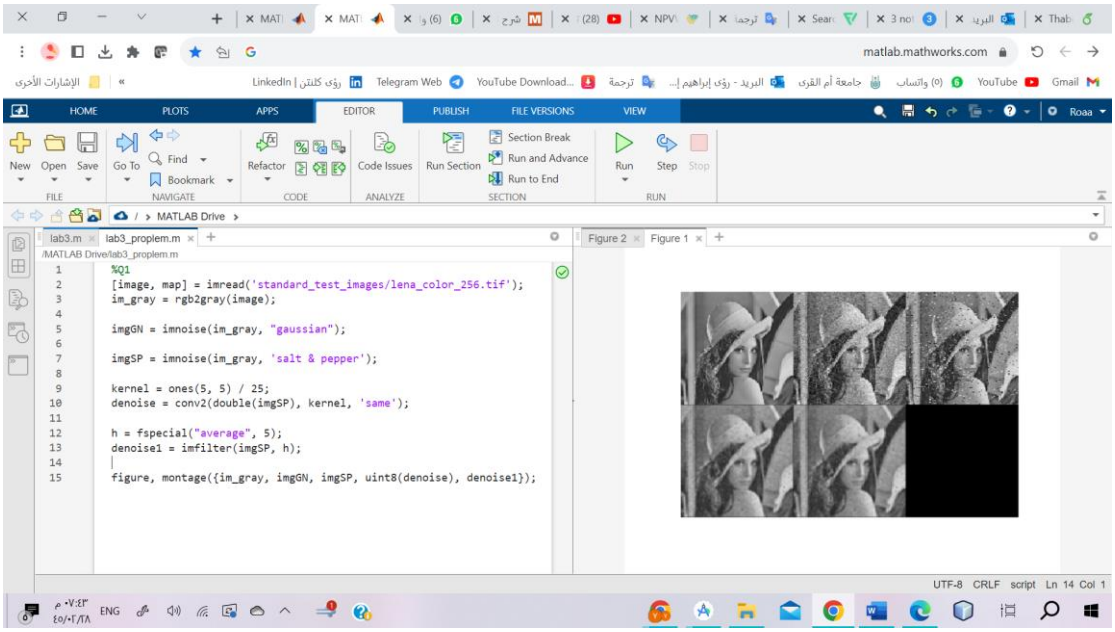



```
[image,map]=imread('standard_test_images/lena_color_256.tif');
%This line reads an image file named
% 'lena_color_256.tif' and stores the image data in the variable image.
% If the image contains an indexed image,
% the colormap will be stored in the variable map.
im_gray=rgb2gray(image);
imshow(im_gray);
%These lines convert the RGB image (image)
% to grayscale using the rgb2gray function and store the result in im_gray.
% Then it displays the grayscale image using the imshow function.

imgGN = imnoise(im_gray, "gaussian");
imshow(imgGN);
%These lines add Gaussian noise to the grayscale image (
% im_gray) using the imnoise function with the noise type specified as
"gaussian".
```

```
% The resulting noisy image is stored in imgGN, and it is displayed using
imshow.

imgSP = imnoise(im_gray, 'salt & pepper');
imshow(imgSP);
%These lines add salt-and-pepper noise
% to the grayscale image (im_gray)
% using the imnoise function with the noise type specified
% as "salt & pepper". The resulting noisy image is stored in imgSP,
% and it is displayed using imshow.

kernel = ones(5, 5) / 25;
denoise = conv2(imgSP, kernel, 'same');
imshow(denoise);
%These lines define a 5x5 averaging kernel (kernel)
% and perform convolution between the noisy image (imgSP)
% and the kernel using the conv2 function.
% The 'same' option ensures that the output size
% is the same as the input size. The resulting denoised image
% is stored in denoise, and it is displayed using imshow.

h = fspecial("average", 5);
denoise1 = imfilter(imgSP, h);
imshow(denoise1);
%These lines create a 5x5 averaging filter
% (h) using the fspecial function and apply
% the filter to the noisy image (imgSP) using the imfilter function.
% The resulting denoised image is stored in denoise1,
% and it is displayed using imshow..

%Overall, this code reads an image,
% converts it to grayscale, adds Gaussian and salt-and-pepper noise,
% performs image denoising using convolution and filtering techniques,
% and displays the intermediate and final results using the imshow
function.

I=imread('standard_test_images/lena_color_512.tif');
%This line reads an image file named 'lena_color_512.tif'
% and stores the image data in the variable I.

I=rgb2gray(I);
%This line converts the RGB image I to grayscale using the rgb2gray
function.

I_noise=imnoise(I,'salt & pepper');
%This line adds salt-and-pepper noise
% to the grayscale image I using the imnoise function.
% The resulting noisy image is stored in the variable I_noise.

mad_lena=medfilt2(I_noise);
%This line applies a median filter to the noisy image I_noise
% using the medfilt2 function.
% The median filtering helps
% in reducing the salt-and-pepper noise.
% The filtered image is stored in the variable mad_lena.

figure,montage({I_noise,mad_lena,I});
%This line creates a figure and displays
% a montage of three images: the noisy image I_noise
```

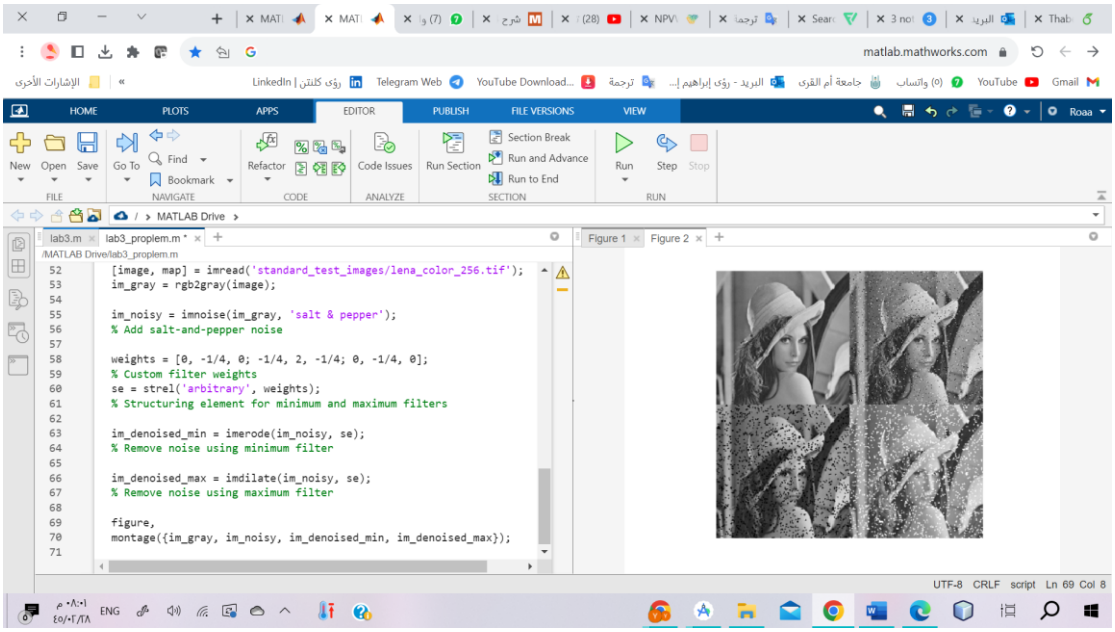
```
% , the filtered image mad_lena,
% and the original grayscale image I.
% The montage function arranges the images
% in a grid for easy comparison.
```

Lab3 proplem:-



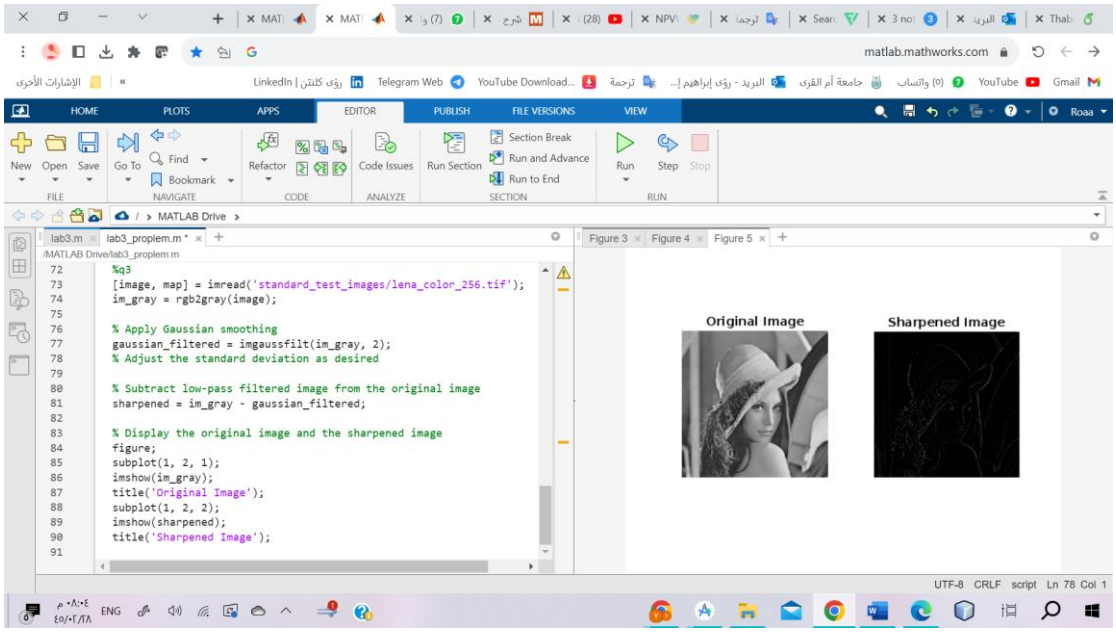

```
%Q1
[image, map] = imread('standard_test_images/lena_color_256.tif');
%This line reads an image file named 'lena_color_256.tif'
% and stores the image data in the variable image.
% If the image contains an indexed image,
% the colormap will be stored in the variable map

im_gray = rgb2gray(image);
%This line converts the RGB image (image) to grayscale
% using the rgb2gray function and stores the result in im_gray.

imgGN = imnoise(im_gray, "gaussian");
%This line adds Gaussian noise to the grayscale image (im_gray)
% using the imnoise function with the noise type specified as "gaussian".
% The resulting noisy image is stored in imgGN.

imgSP = imnoise(im_gray, 'salt & pepper');
%This line adds salt-and-pepper noise
% to the grayscale image (im_gray)
% using the imnoise function with the noise type specified
% as "salt & pepper". The resulting noisy image is stored in imgSP.

kernel = ones(5, 5) / 25;
denoise = conv2(double(imgSP), kernel, 'same');
%These lines create a 5x5 averaging kernel (kernel),
% convert the noisy image (imgSP) to a 2D array of type double,
```

```
% and perform convolution between the noisy image and the kernel
% using the conv2 function. T
% he 'same' option ensures that the output size
% is the same as the input size.
% The resulting denoised image is stored in denoise.

h = fspecial("average", 5);
denoise1 = imfilter(imgSP, h);
%These lines create a 5x5 averaging filter
% (h) using the fspecial function
% and apply the filter to the noisy image (imgSP)
% using the imfilter function.
% The resulting denoised image is stored in denoise1.

figure, montage({im_gray, imgGN, imgSP, uint8(denoise), denoise1});
%This line creates a new figure
% and displays a montage of images for comparison.
% The montage includes the original grayscale image (im_gray),
% the image with added Gaussian noise (imgGN),
% the image with added salt-and-pepper noise (imgSP),
% the denoised image obtained through convolution (denoise),
% and the denoised image obtained through filtering (denoise1).
```




```
%q2
[image, map] = imread('standard_test_images/lena_color_256.tif');
im_gray = rgb2gray(image);

im_noisy = imnoise(im_gray, 'salt & pepper');
% Add salt-and-pepper noise

weights = [0, -1/4, 0; -1/4, 2, -1/4; 0, -1/4, 0];
% Custom filter weights
se = strel('arbitrary', weights);
```

```
% Structuring element for minimum and maximum filters

im_denoised_min = imerode(im_noisy, se);
% Remove noise using minimum filter

im_denoised_max = imdilate(im_noisy, se);
% Remove noise using maximum filter

figure,
montage({im_gray, im_noisy, im_denoised_min, im_denoised_max});
```




```
%q3
[image, map] = imread('standard_test_images/lena_color_256.tif');
im_gray = rgb2gray(image);

% Apply Gaussian smoothing
gaussian_filtered = imgaussfilt(im_gray, 2);
% Adjust the standard deviation as desired

% Subtract low-pass filtered image from the original image
sharpened = im_gray - gaussian_filtered;

% Display the original image and the sharpened image
figure;
subplot(1, 2, 1);
imshow(im_gray);
title('Original Image');
subplot(1, 2, 2);
imshow(sharpened);
title('Sharpened Image');
```

رؤى إبراهيم كلنتن 441004834
معالجة صور شعبة نظري 1 شعبة العملي 4