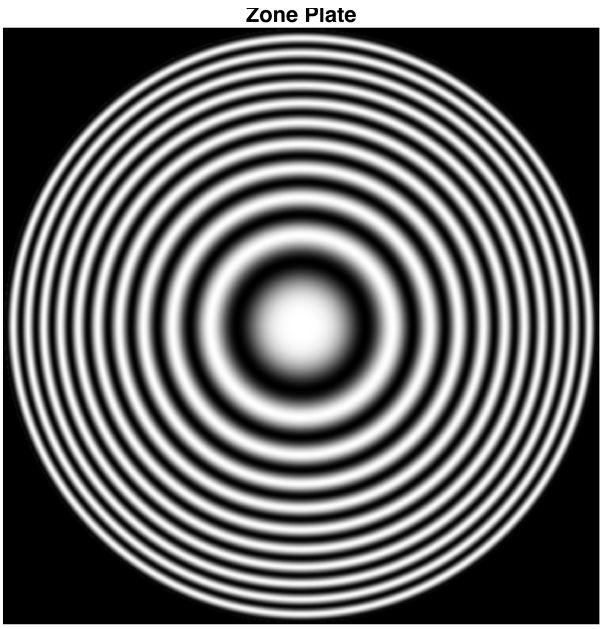
Assignment #2:

Name: Aishwarya Dekhane - adekhane@umich.edu

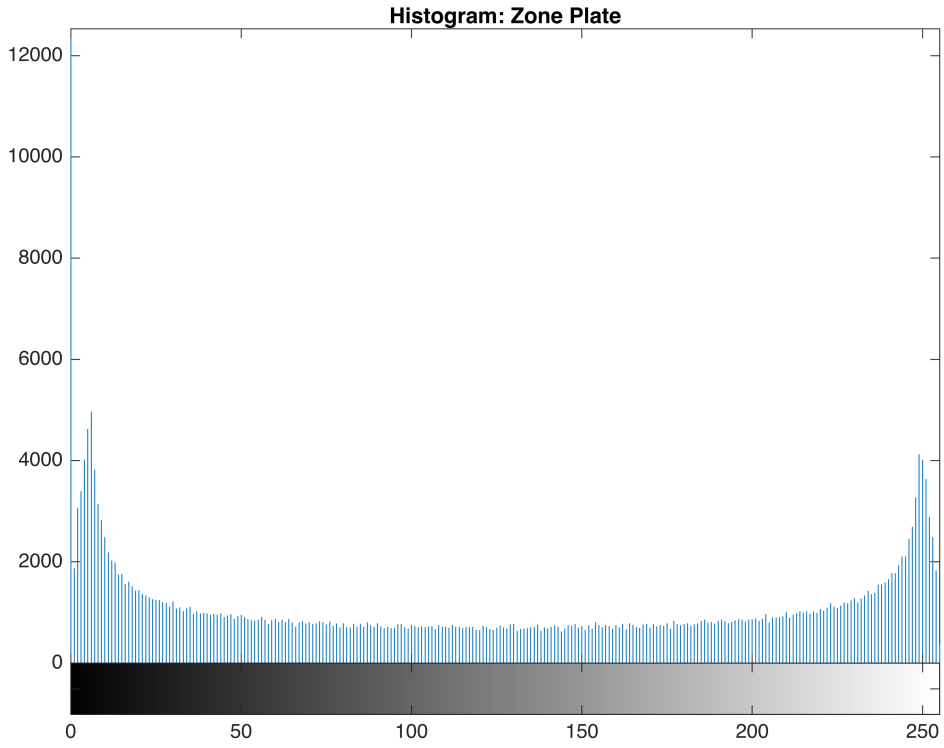
The purpose of this assignment is to show how to construct a high pass, bandpass, and bandreject filter using only low pass filters. Then, it is possible to construct 2D filters from a 1D filter, and to use a low pass filter to compensate for uneven intensity in an image. Filter design software will perform all these operations, but it is useful to know how this is performed.

1. Load the image "zoneplate.tif" and display the image and it's histogram

```
image = imread('zoneplate.tif');
figure;
imshow(image);
title('Zone Plate');
```




```
figure;
imhist(image);
title('Histogram: Zone Plate');
```



2. What is the range of values present in the image?

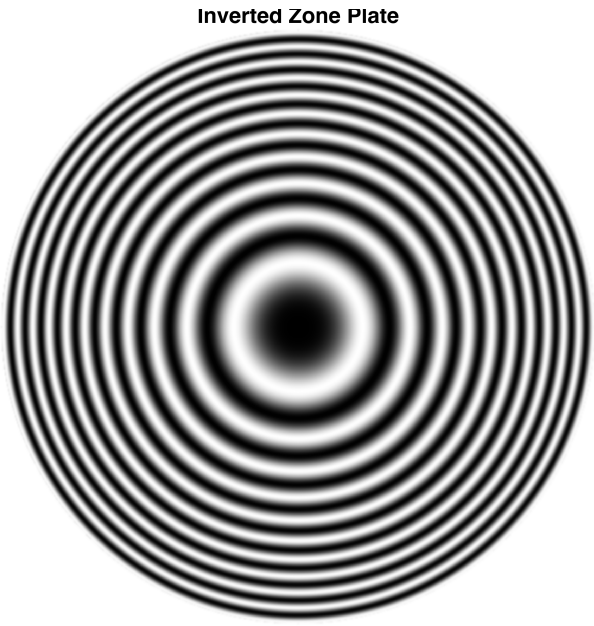
Answer: The histogram depicts pixel intensity values ranging from 0 to approximately 255 along the x-axis.

The y-axis shows the frequency of pixels possessing a specific intensity value, indicating how often each intensity level occurs in the image.

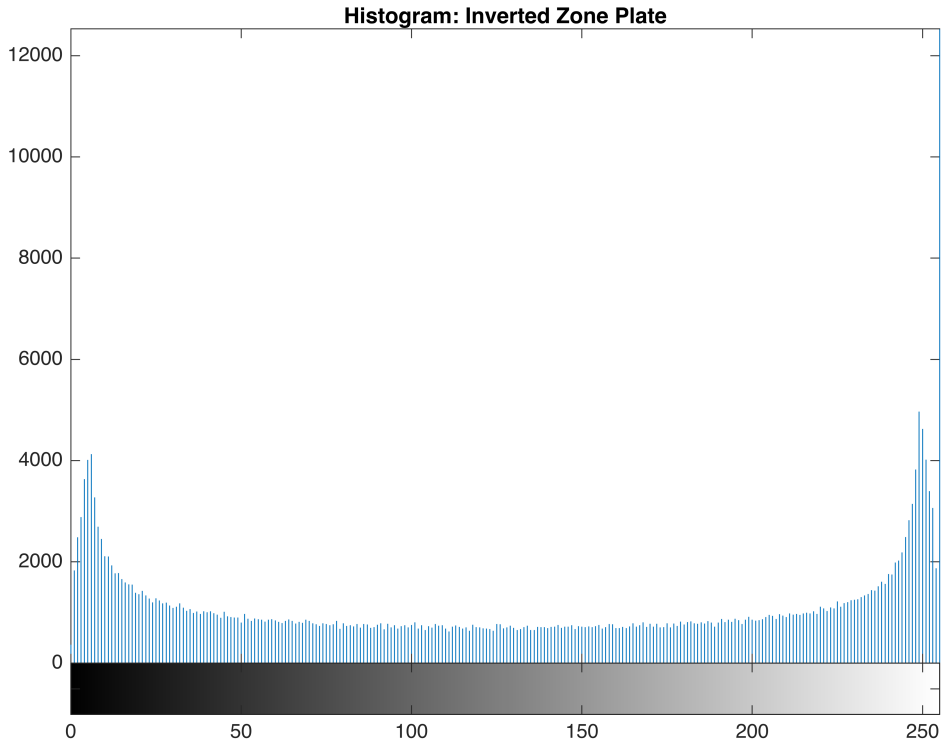
A value of 0 corresponds to black, while 255 corresponds to white.

3. Invert the image and print the inverted image and it's histrogram

```
inverted_img = 255 - image;
figure;
imshow(inverted_img);
title('Inverted Zone Plate');
```




```
figure;
imhist(inverted_img);
title('Histogram: Inverted Zone Plate');
```

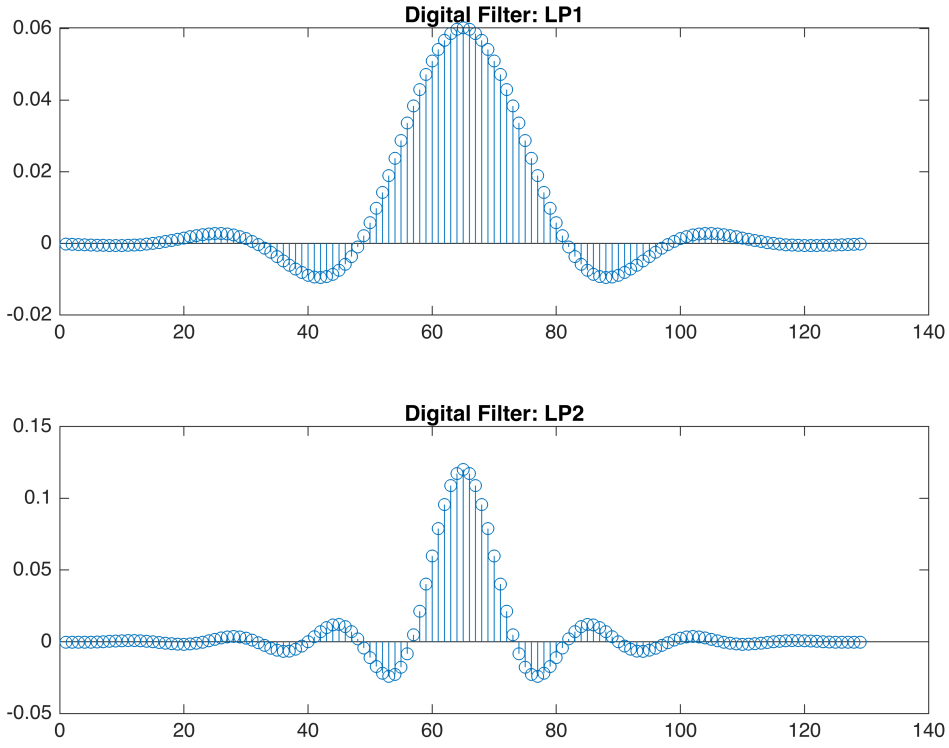


The following will read in the coefficients of digital filters. These were designed using MATLAB's Signal Processing Toolbox and the fir1 function, so these are finite impulse response filters (and therefore stable).

```
lp1=dlmread('oneDlowpass1.txt');
lp2=dlmread('oneDlowpass2.txt');
```

4. (10 points) Plot both lp1 and lp2.

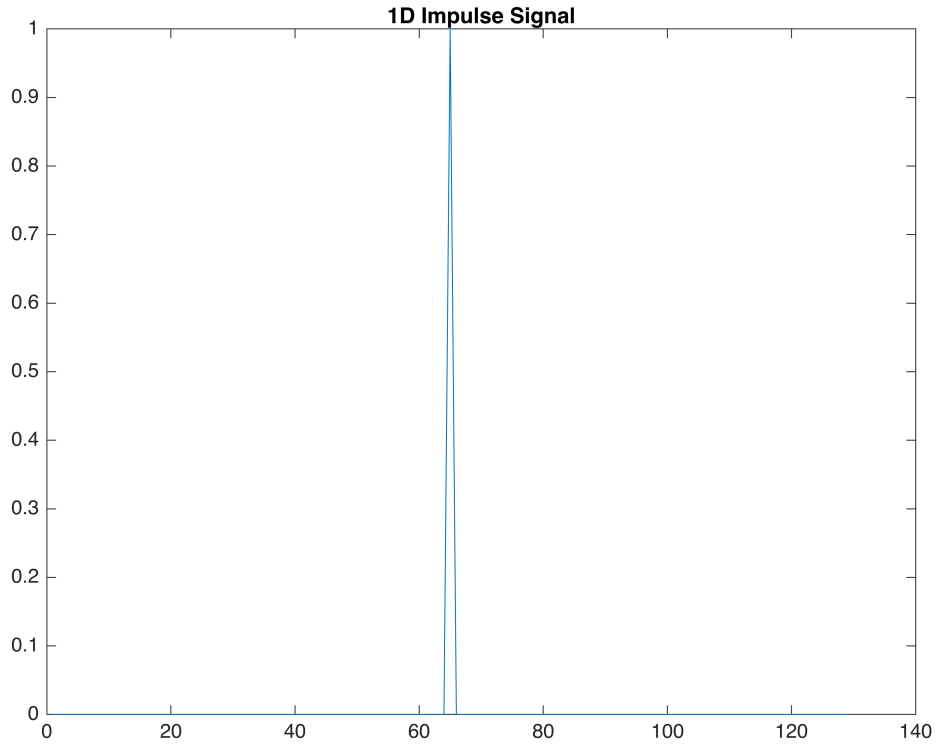
```
% Plot both filters
figure;
subplot(2,1,1);
stem(lp1);
title('Digital Filter: LP1');
subplot(2,1,2);
stem(lp2);
title('Digital Filter: LP2');
```



The following code will generate an impulse at the center of the length of the sequence. Note that the Fourier Transform of a unit impulse is the constant 1.

```
M_el = numel(lp1);
center = ceil(M_el / 2);
oneDimpulse = zeros(1, M_el);
oneDimpulse(center) = 1;
```

```
figure;
plot(oneDimpulse);
title('1D Impulse Signal');
```



There are four main categories of filters:

1. low pass
2. high pass
3. band reject (notch)
4. band pass

These can all be generated from a low pass filter and an impulse of the the correct size:

High Pass: HP = Impulse - LP

Band Reject: BR = LP + HP

Since the LP will have a lower "cutoff frequency" than the high pass filter, this can be written as (HP2 just indicates the cutoff frequency used is different from LP1):

Band Reject = LP1 + HP2

Band Pass = Impulse - Band Reject

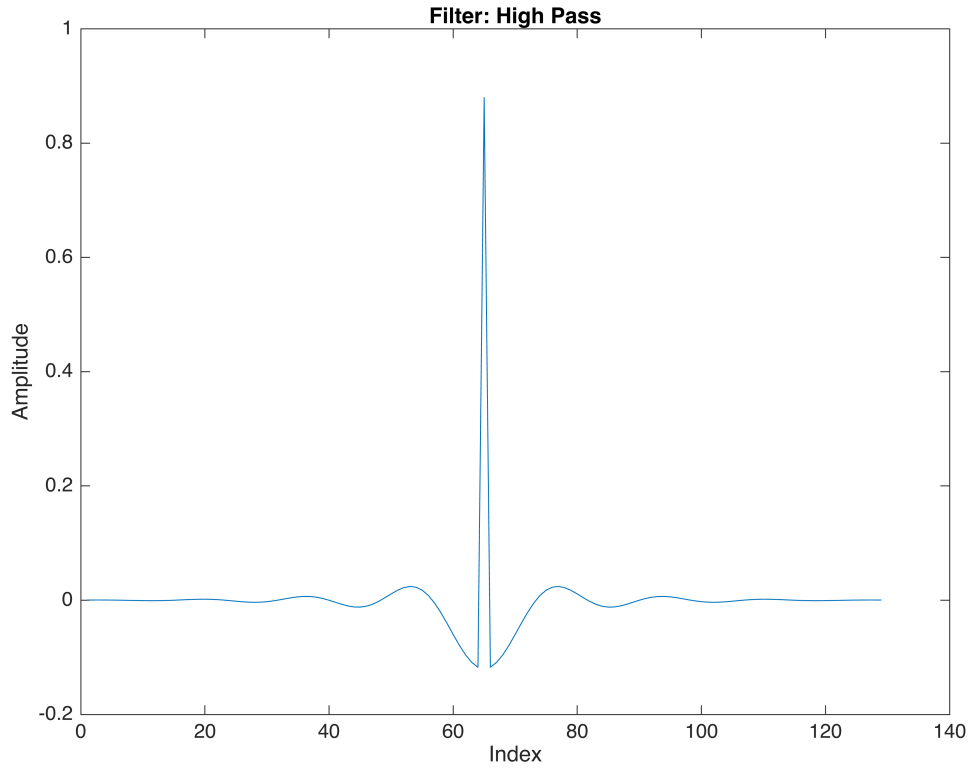
5. (10 points) Generate, using the given lp1 and lp2, the following filters: the impulse used must be the same length as the low pass filter used, in this case the oneDimpulse has the correct length.

a. a high pass filter using lp2

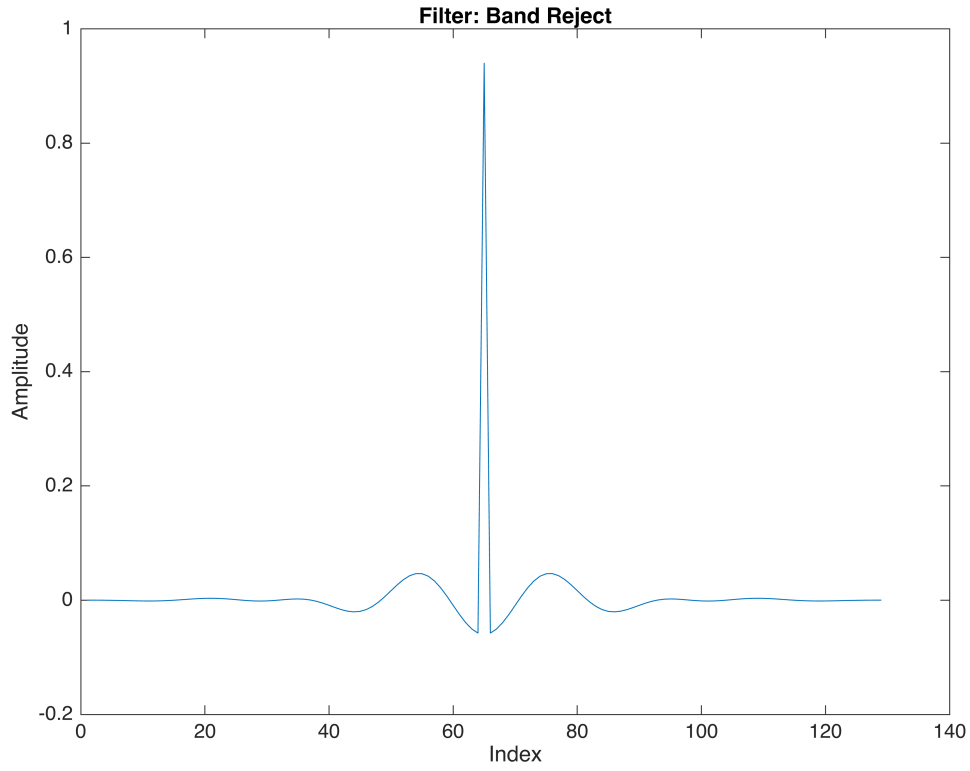
b. a band reject filter

c. a band pass filter

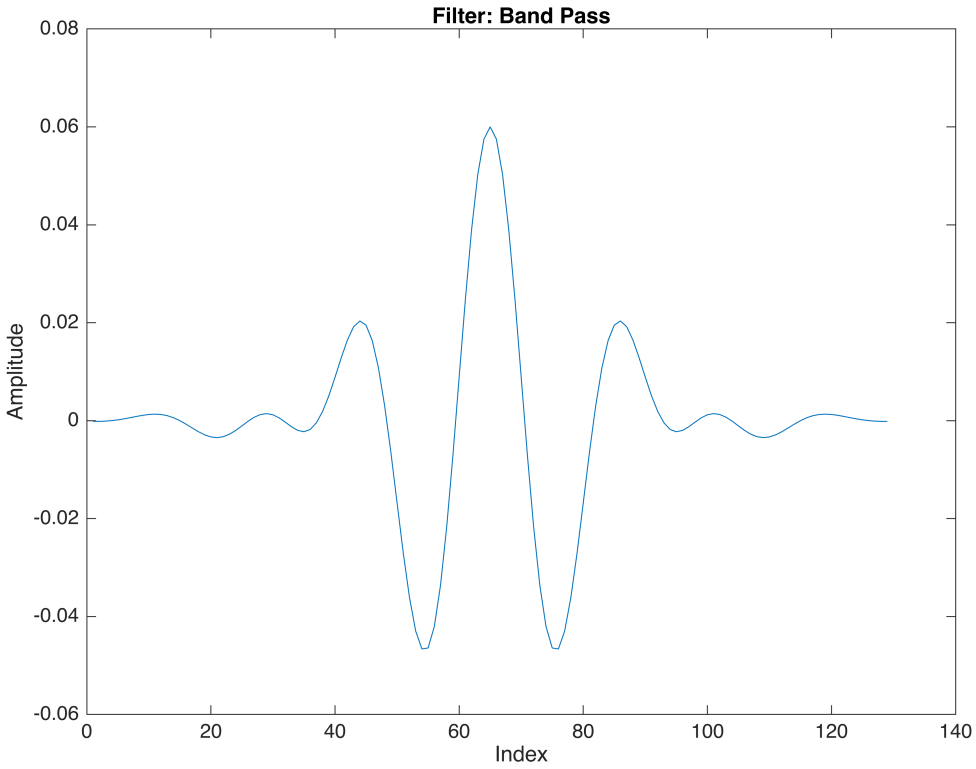
```
HP = oneDimpulse - lp2;
figure;
plot(HP);
title('Filter: High Pass');
xlabel('Index');
ylabel('Amplitude');
```




```
BR = lp1 + HP;
figure;
plot(BR);
title('Filter: Band Reject');
xlabel('Index');
ylabel('Amplitude');
```



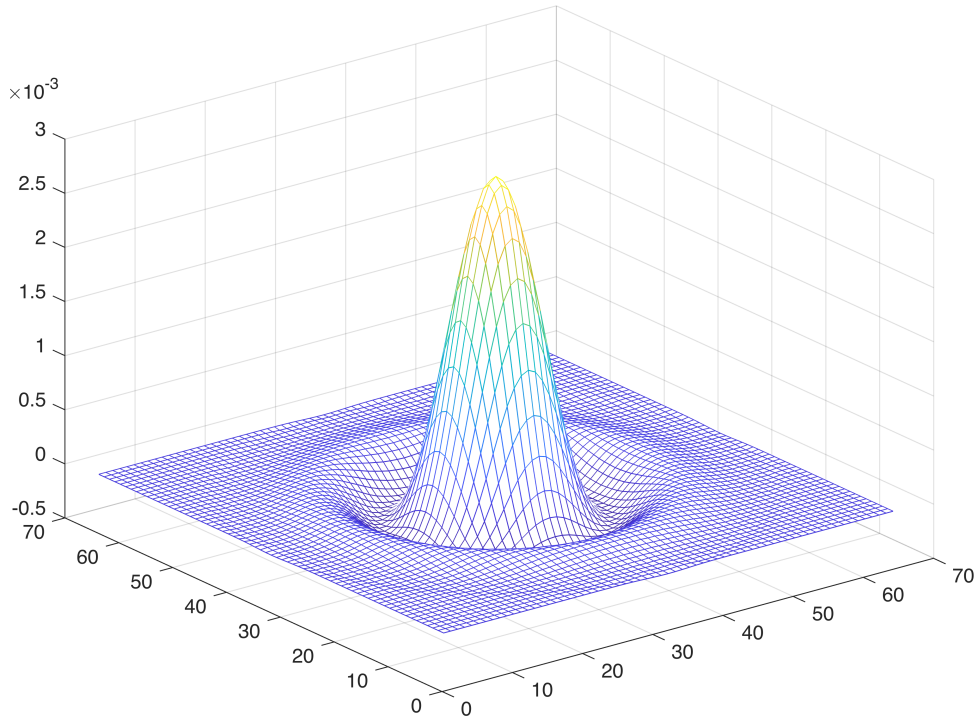

```
BP = oneDimpulse - BR;
figure;
plot(BP);
title('Filter: Band Pass');
xlabel('Index');
ylabel('Amplitude');
```



6. (10 points) There are several ways to generate a 2D filter from a given 1D filter. It's possible to just generate a square matrix by multiplying the two 1D filters together. This will work but will often generate ringing, so a windowing method is used to smooth the transition between edges of the image. MATLAB has a built in function that makes this easy: ftrans2

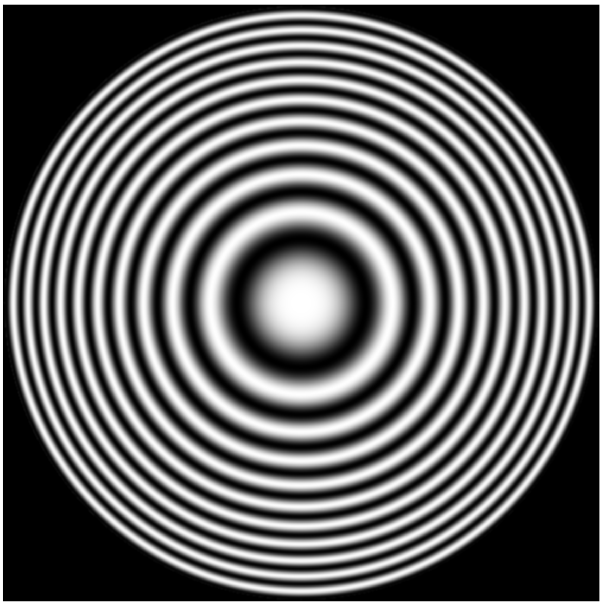
As an example: to generate a 2D low pass filter using the coefficients of the filter given in lp1:

```
LP1_2D=ftrans2(lp1);
mesh(LP1_2D(1:2:end,1:2:end))
```

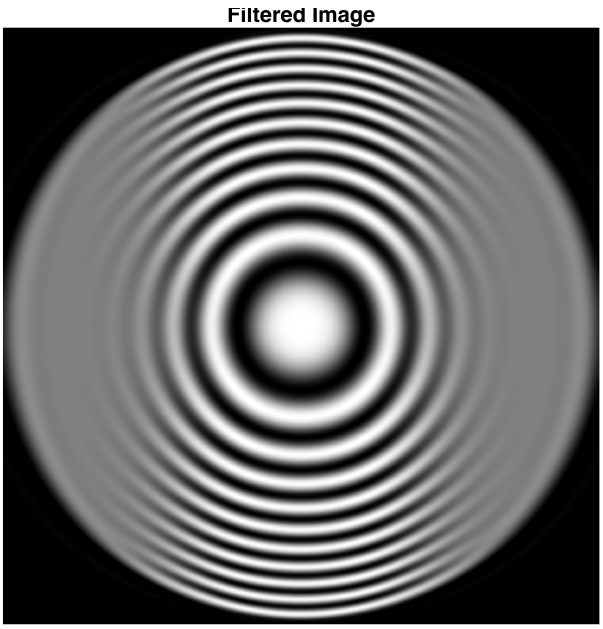


Now use this to lowpassfilter the zoneplate image using the imfilter MATLAB command:

```
filtered_img = imfilter(image, lp1);
figure;
imshow(image);
```



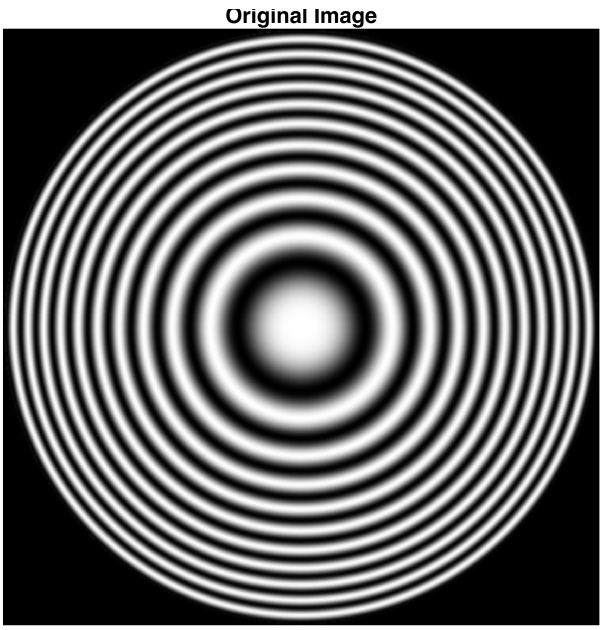
```
figure;
imshow(filtered_img);
title('Filtered Image');
```



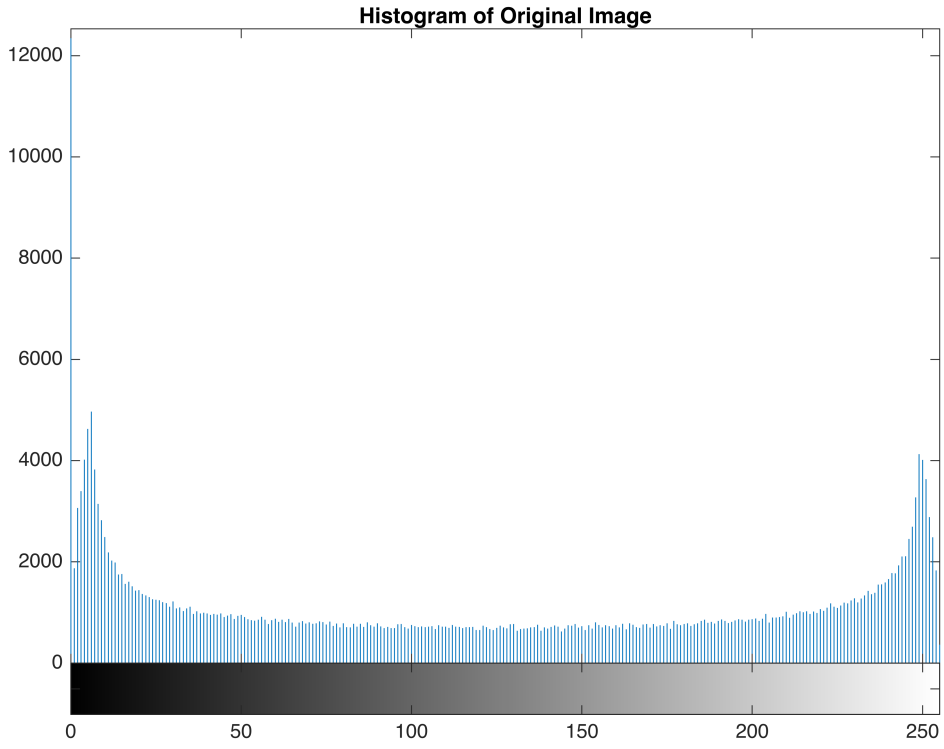
Plot the filters generated in 5. Apply the three filters generated to the zoneplate image, plot the results, and plot the histogram of each result.

```
% Apply the filters to the zoneplate image
filtered_HP = imfilter(image, HP);
filtered_BR = imfilter(image, BR);
filtered_BP = imfilter(image, BP);

% Display the original image and its histogram
figure;
imshow(image);
title('Original Image');
```

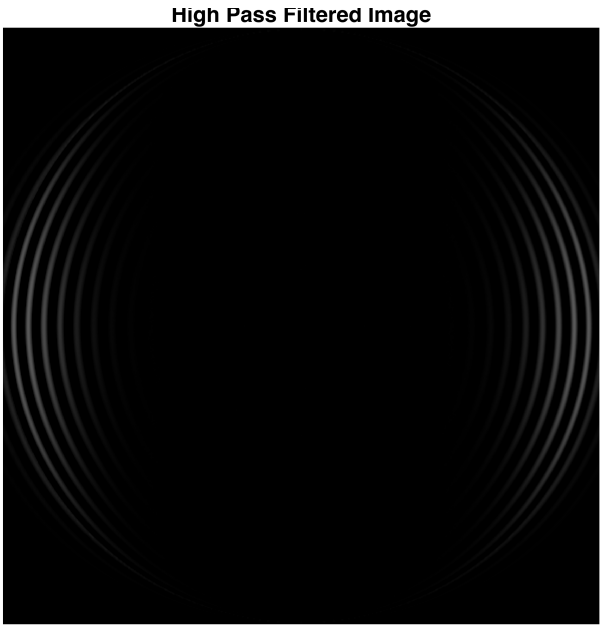



```
figure;
imhist(image);
title('Histogram of Original Image');
```

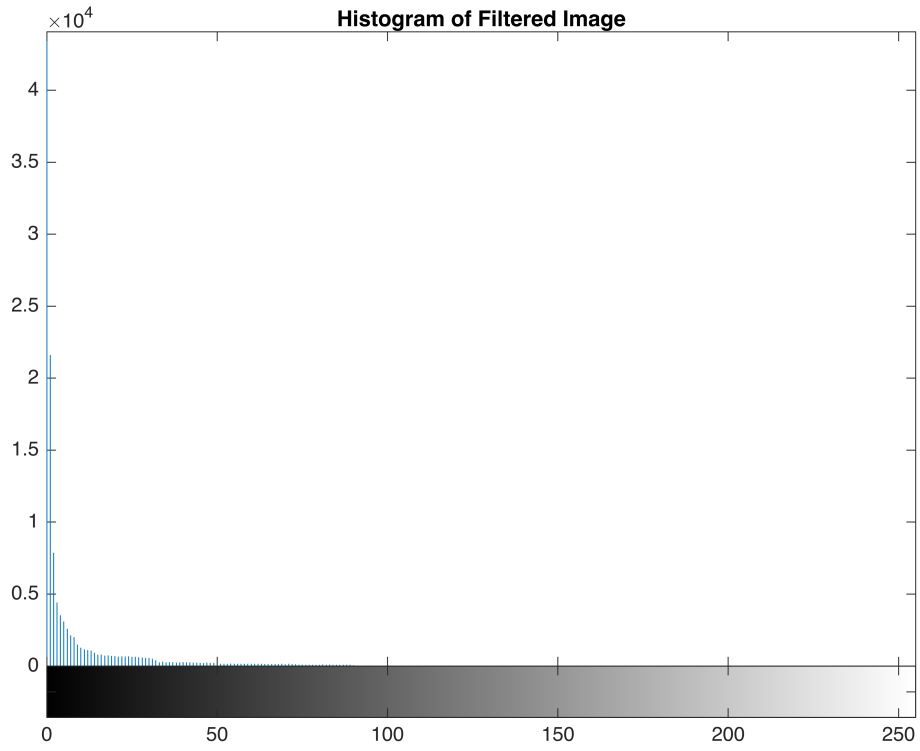



```
% Display the filtered image (High Pass) and its histogram
```

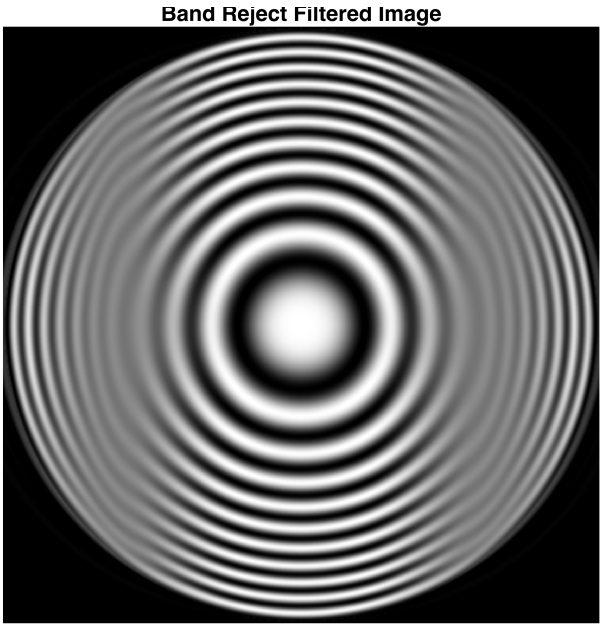
```
figure;
imshow(filtered_HP);
title('High Pass Filtered Image');
```



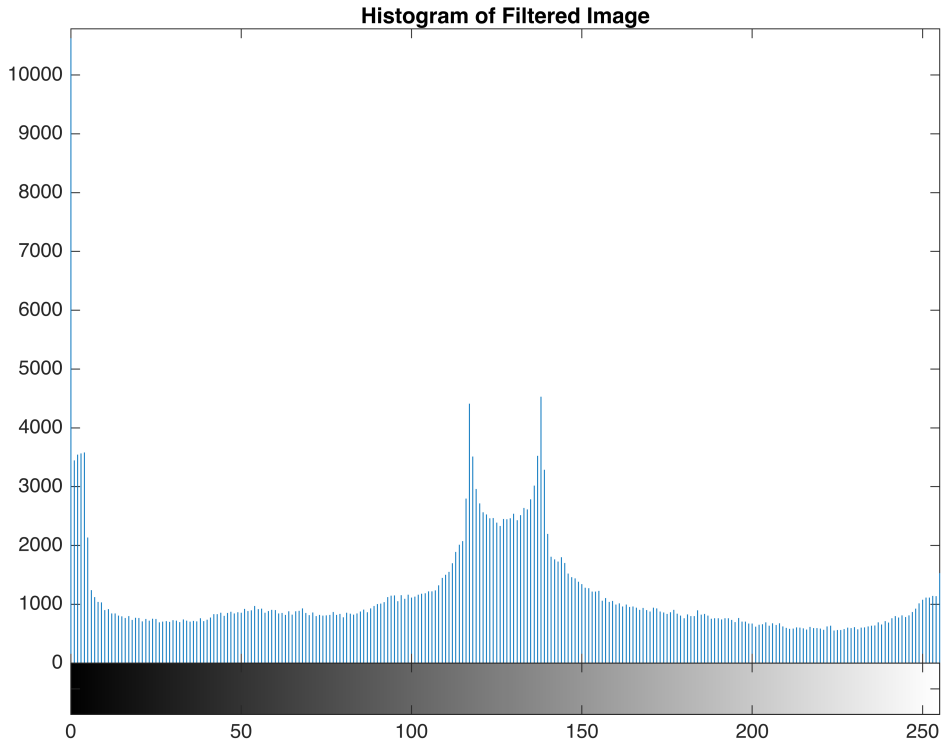

```
figure;
imhist(filtered_HP);
title('Histogram of Filtered Image');
```



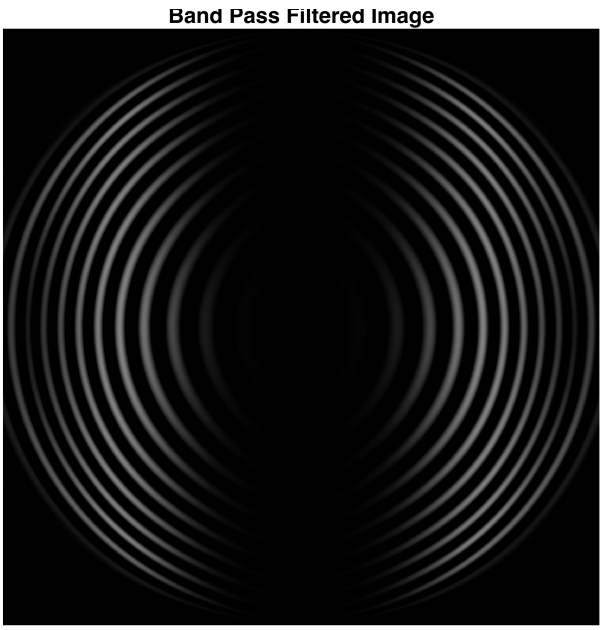

```
% Display the filtered image (Band Reject) and its histogram
figure;
imshow(filtered_BR);
title('Band Reject Filtered Image');
```



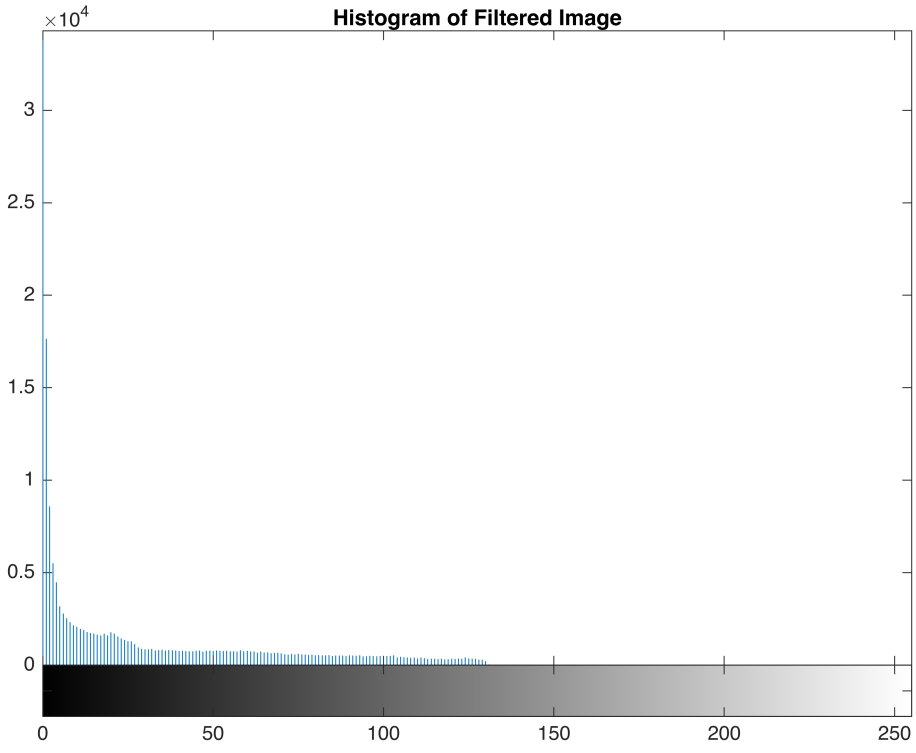
```
figure;
imhist(filtered_BR);
title('Histogram of Filtered Image');
```




```
% Display the filtered image (Band Pass) and its histogram
figure;
imshow(filtered_BP);
title('Band Pass Filtered Image');
```

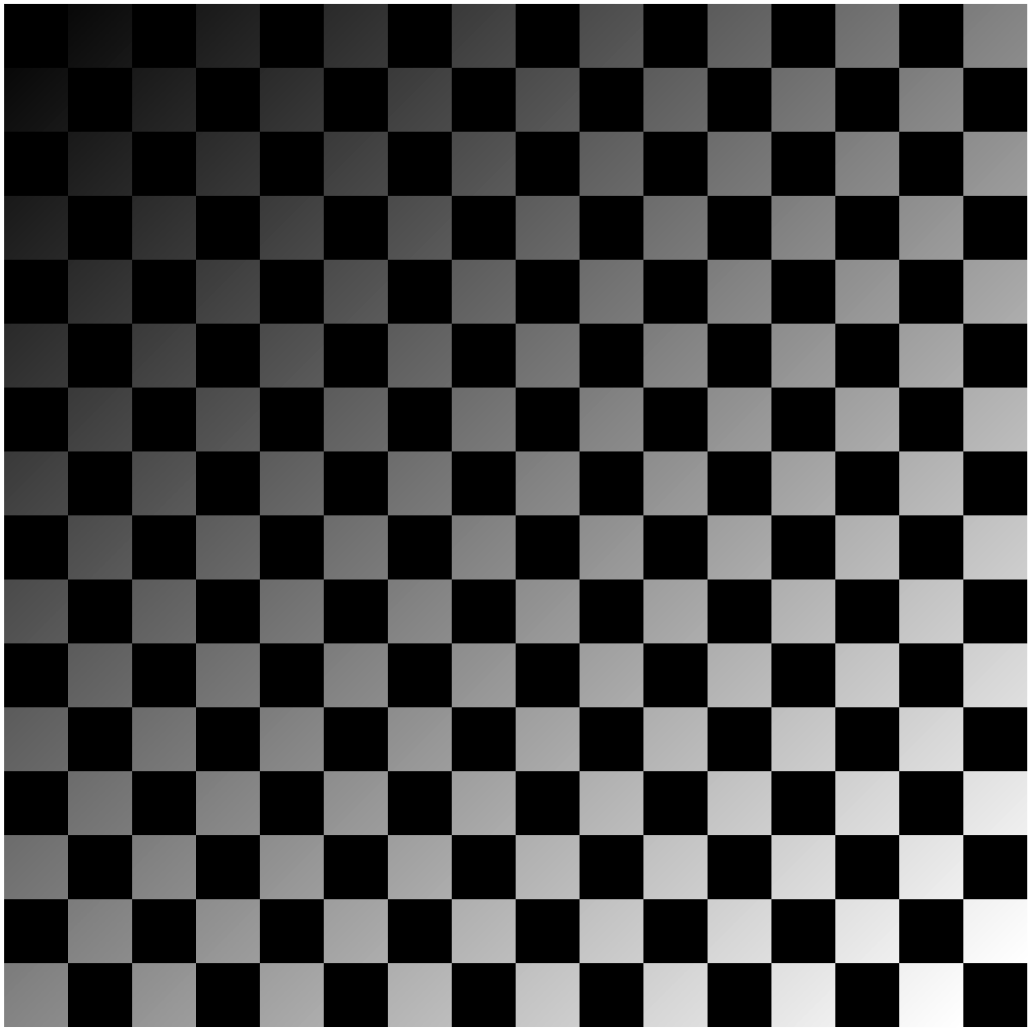



```
figure;
imhist(filtered_BP);
title('Histogram of Filtered Image');
```



Low pass filtering can be used to compensate for uneven illumation. For example, the following checkerboard image has different shading, showing that the illumination is not consistent. This code reads in the image, converts to double, then rescales the image to the range [0,1], to make computations take place in the same range.

```
cb_shaded = im2double(imread("checkerboard1024-shaded.tif"));
cb_shaded = rescale(cb_shaded, 0, 1);
imshow(cb_shaded)
```

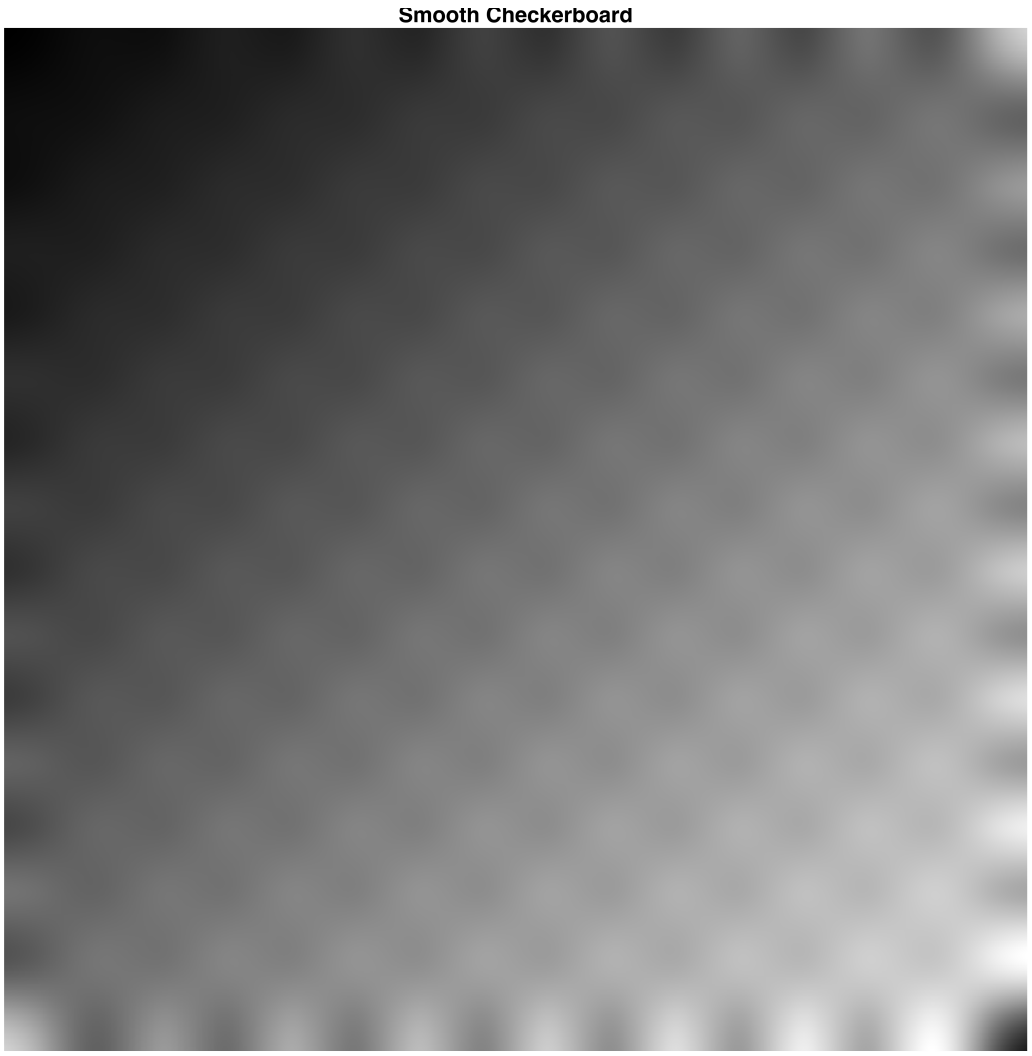


7a. (10 points) Use a Gaussian filter to blur this image. The result should be a consistent image that varies in instensity smoothly, with no evidence of the individual squares. Use the imgaussfilt function to perform the blurring operationa and experiment with the value of sigma until the result is a smooth image.

Use the imshow command, with the [] option, to display your result and save the result in a variable.

```
smooth_img = imgaussfilt(cb_shaded, 40);
figure;
```

```
imshow(smooth_img, []);
title('Smooth Checkerboard')
```



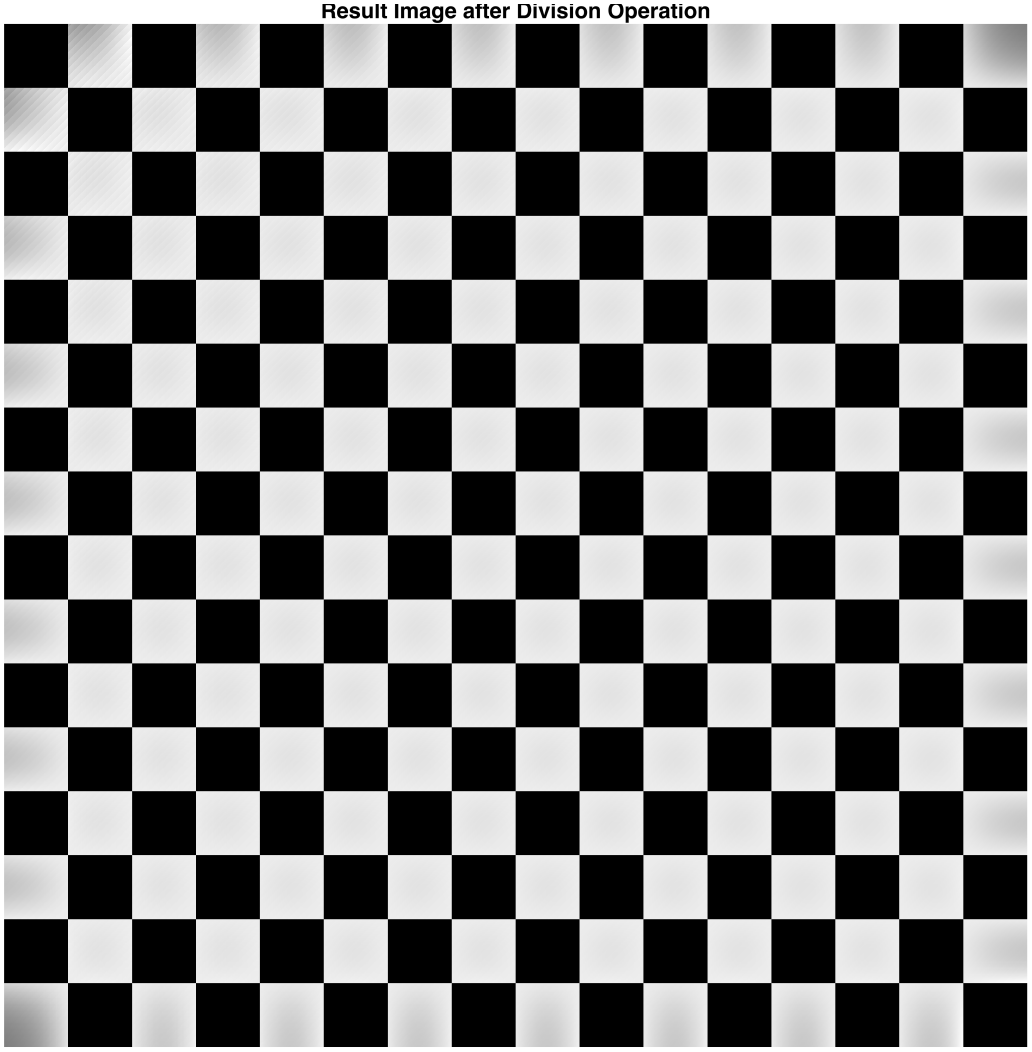
7.b. (10 points) Now divide the original image by the the result in 7a. Use the "elementwise" division, which in MATLAB is the ./ command

Plot the results with the imshow command. It will be useful to use the [] option in imshow to scale the output.

Comment on the output. With a correct amount of blur, the result should be a checkerboard pattern that does not have the shading strongly shown. Does the result show a perfectly even checkerboard ? Expalin using the intensity level from 7a, is that completely smoorth or does some variation exist? Why ?

```
result_img = cb_shaded ./ smooth_img;
figure;
imshow(result_img, []);
title('Result Image after Division Operation');
```

Result Image after Division Operation



Comment:

Even with a Gaussian Blur filter set at a sigma value, there may still be subtle fluctuations present in the blurred image. While adjusting the sigma parameter does impact the level of smoothing, the intricate details of the checkboard pattern introduce complexities that cannot be completely eliminated, leaving residual variations in the smoothed image. These variations stem from the inherent method of pixel averaging employed by the Gaussian Blur filter, which inherently retains some level of variation within the image.