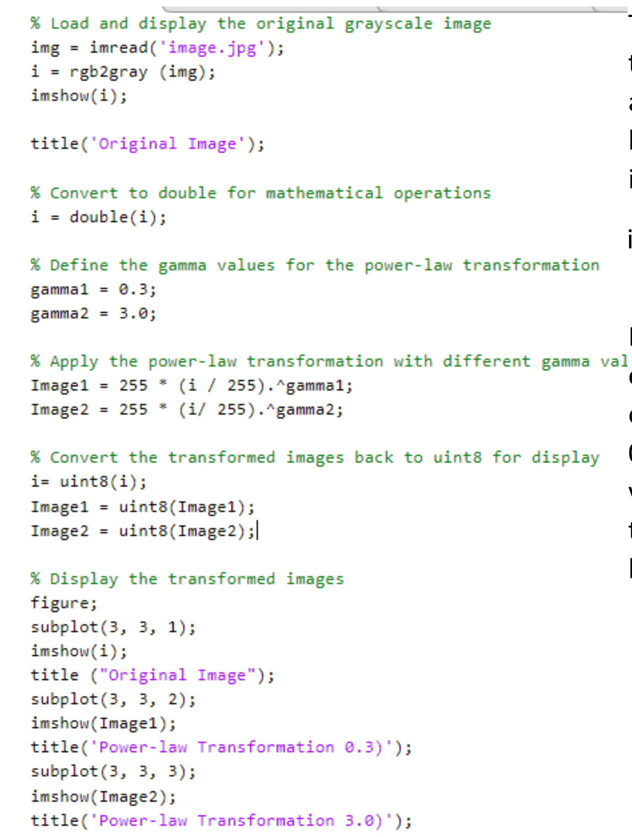
Problem 1:



```
% Load and display the original grayscale image
img = imread('image.jpg');
i = rgb2gray (img);
imshow(i);

title('Original Image');

% Convert to double for mathematical operations
i = double(i);

% Define the gamma values for the power-law transformation
gamma1 = 0.3;
gamma2 = 3.0;

% Apply the power-law transformation with different gamma val
Image1 = 255 * (i / 255).^gamma1;
Image2 = 255 * (i/ 255).^gamma2;

% Convert the transformed images back to uint8 for display
i= uint8(i);
Image1 = uint8(Image1);
Image2 = uint8(Image2);

% Display the transformed images
figure;
subplot(3, 3, 1);
imshow(i);
title ("Original Image");
subplot(3, 3, 2);
imshow(Image1);
title('Power-law Transformation 0.3)');
subplot(3, 3, 3);
imshow(Image2);
title('Power-law Transformation 3.0)');
```

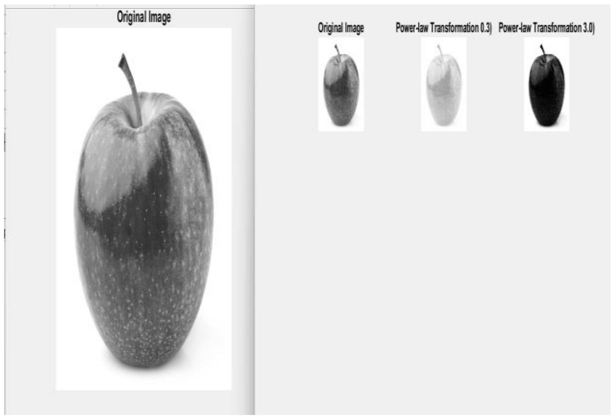


Figure 1

i. Log Transformation

$$s = c\log(1 + r)$$

The image's dark sections are made lighter by the log transformation, which also expands the low-intensity regions and compresses the high-intensity ones.

ii. Inverse Log Transformation

$$s = c\log^{-1}(r)$$

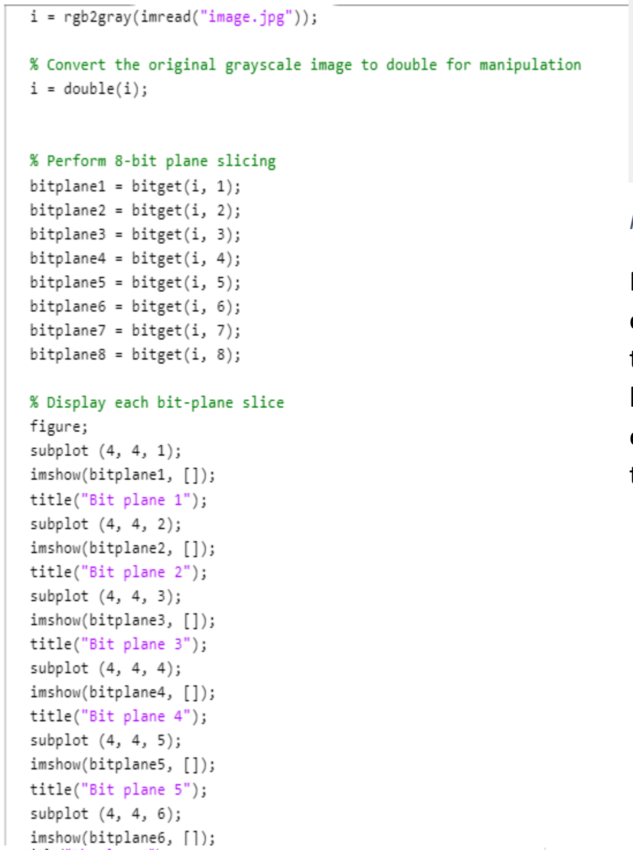
The image is made darker by the inverse log transformation. The inverse of log transformation is accomplished. It distorts the image by enlarging the high-intensity regions and contracting the low-intensity ones.

iii. Power Law transformation

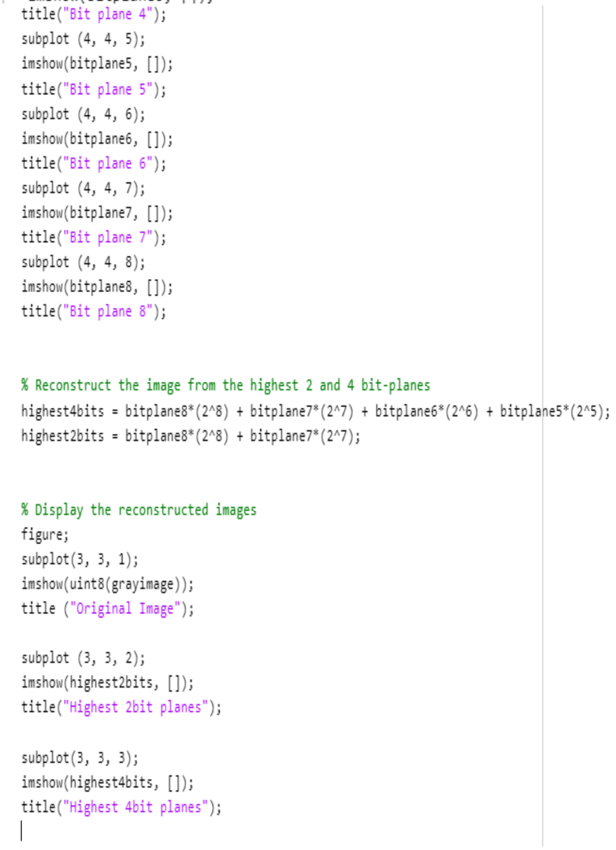
$$s = c\,r^{\gamma}$$

Figure 1 displays the two versions of the image: the one before the power law transformation and the one after. The images in the figure have gammas of 0.3 and 3. The graphic shows that when the gamma value is less than 1, it behaves like a log transformation, and when it is more than 1, it behaves like an inverse log transformation.

Problem 2:



```
i = rgb2gray(imread("image.jpg"));

% Convert the original grayscale image to double for manipulation
i = double(i);


% Perform 8-bit plane slicing
bitplane1 = bitget(i, 1);
bitplane2 = bitget(i, 2);
bitplane3 = bitget(i, 3);
bitplane4 = bitget(i, 4);
bitplane5 = bitget(i, 5);
bitplane6 = bitget(i, 6);
bitplane7 = bitget(i, 7);
bitplane8 = bitget(i, 8);

% Display each bit-plane slice
figure;
subplot (4, 4, 1);
imshow(bitplane1, []);
title("Bit plane 1");
subplot (4, 4, 2);
imshow(bitplane2, []);
title("Bit plane 2");
subplot (4, 4, 3);
imshow(bitplane3, []);
title("Bit plane 3");
subplot (4, 4, 4);
imshow(bitplane4, []);
title("Bit plane 4");
subplot (4, 4, 5);
imshow(bitplane5, []);
title("Bit plane 5");
subplot (4, 4, 6);
imshow(bitplane6, []);
```



```
title("Bit plane 4");
subplot (4, 4, 5);
imshow(bitplane5, []);
title("Bit plane 5");
subplot (4, 4, 6);
imshow(bitplane6, []);
```

```
title("Bit plane 6");
subplot (4, 4, 7);
imshow(bitplane7, []);
title("Bit plane 7");
subplot (4, 4, 8);
imshow(bitplane8, []);
title("Bit plane 8");



% Reconstruct the image from the highest 2 and 4 bit-planes
highest4bits = bitplane8*(2^8) + bitplane7*(2^7) + bitplane6*(2^6) + bitplane5*(2^5);
highest2bits = bitplane8*(2^8) + bitplane7*(2^7);



% Display the reconstructed images
figure;
subplot(3, 3, 1);
imshow(uint8(grayimage));
title ("Original Image");

subplot (3, 3, 2);
imshow(highest2bits, []);
title("Highest 2bit planes");

subplot(3, 3, 3);
imshow(highest4bits, []);
title("Highest 4bit planes");
```

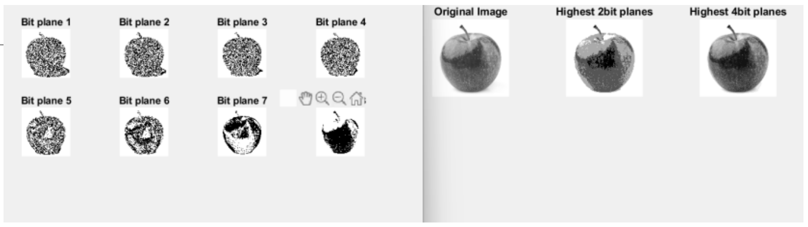
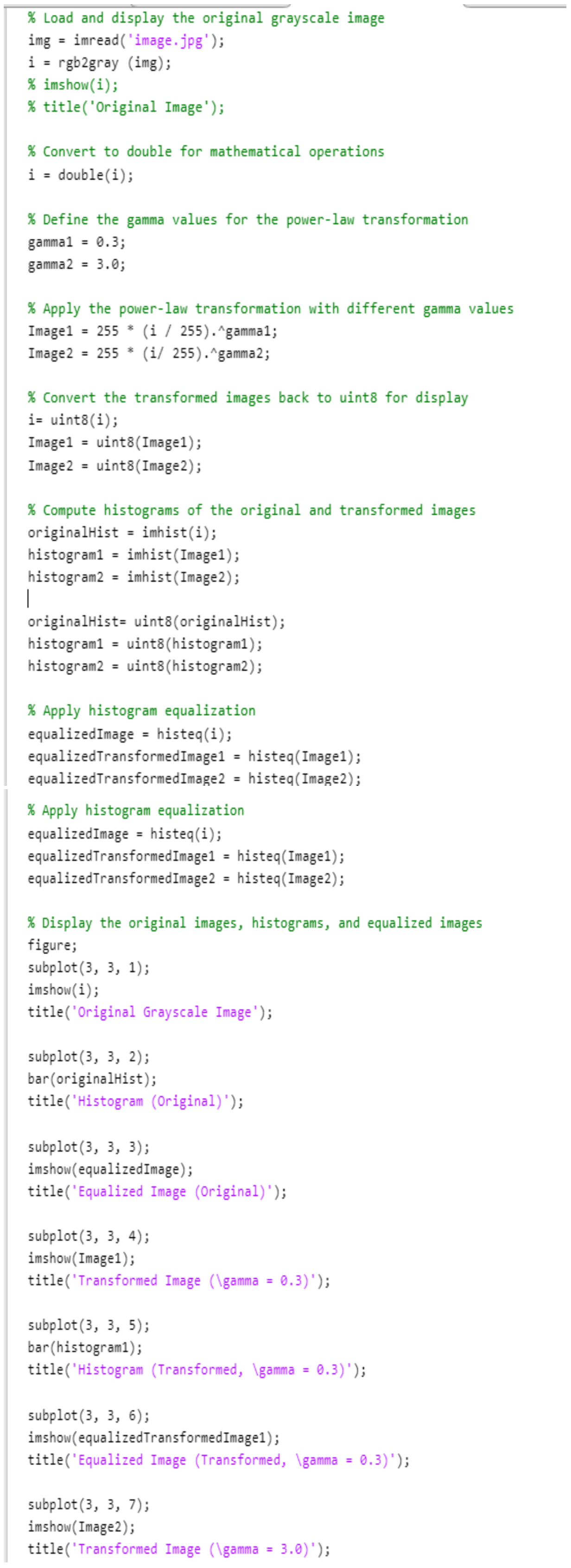


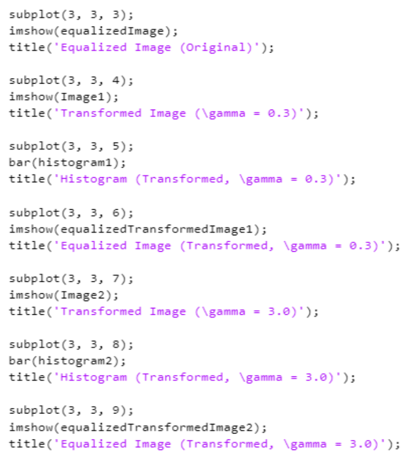
Figure 2

Figure 2 shows that the 8-bit place slicing has occurred. We can also see the original image and the reconstructed image with the highest 2-bit and highest 4-bit on the left pane of figure 2. It is observed that the 4-bit image has more detail than the 2-bit image and is more pronounced and darker.

Problem 3:



```
% Load and display the original grayscale image
img = imread('image.jpg');
i = rgb2gray (img);
% imshow(i);
% title('Original Image');

% Convert to double for mathematical operations
i = double(i);

% Define the gamma values for the power-law transformation
gamma1 = 0.3;
gamma2 = 3.0;

% Apply the power-law transformation with different gamma values
Image1 = 255 * (i / 255).^gamma1;
Image2 = 255 * (i/ 255).^gamma2;

% Convert the transformed images back to uint8 for display
i= uint8(i);
Image1 = uint8(Image1);
Image2 = uint8(Image2);

% Compute histograms of the original and transformed images
originalHist = imhist(i);
histogram1 = imhist(Image1);
histogram2 = imhist(Image2);

originalHist= uint8(originalHist);
histogram1 = uint8(histogram1);
histogram2 = uint8(histogram2);

% Apply histogram equalization
equalizedImage = histeq(i);
equalizedTransformedImage1 = histeq(Image1);
equalizedTransformedImage2 = histeq(Image2);
% Apply histogram equalization
equalizedImage = histeq(i);
equalizedTransformedImage1 = histeq(Image1);
equalizedTransformedImage2 = histeq(Image2);

% Display the original images, histograms, and equalized images
figure;
subplot(3, 3, 1);
imshow(i);
title('Original Grayscale Image');

subplot(3, 3, 2);
bar(originalHist);
title('Histogram (Original)');

subplot(3, 3, 3);
imshow(equalizedImage);
title('Equalized Image (Original)');

subplot(3, 3, 4);
imshow(Image1);
title('Transformed Image (\gamma = 0.3)');

subplot(3, 3, 5);
bar(histogram1);
title('Histogram (Transformed, \gamma = 0.3)');

subplot(3, 3, 6);
imshow(equalizedTransformedImage1);
title('Equalized Image (Transformed, \gamma = 0.3)');

subplot(3, 3, 7);
imshow(Image2);
title('Transformed Image (\gamma = 3.0)');
```



```
subplot(3, 3, 3);
imshow(equalizedImage);
title('Equalized Image (Original)');

subplot(3, 3, 4);
imshow(Image1);
title('Transformed Image (\gamma = 0.3)');

subplot(3, 3, 5);
bar(histogram1);
title('Histogram (Transformed, \gamma = 0.3)');

subplot(3, 3, 6);
imshow(equalizedTransformedImage1);
title('Equalized Image (Transformed, \gamma = 0.3)');

subplot(3, 3, 7);
imshow(Image2);
title('Transformed Image (\gamma = 3.0)');

subplot(3, 3, 8);
bar(histogram2);
title('Histogram (Transformed, \gamma = 3.0)');

subplot(3, 3, 9);
imshow(equalizedTransformedImage2);
title('Equalized Image (Transformed, \gamma = 3.0)');
```

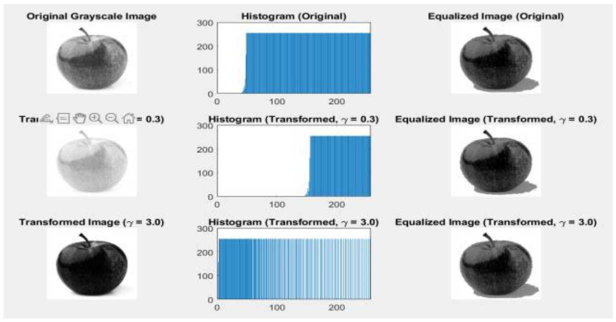
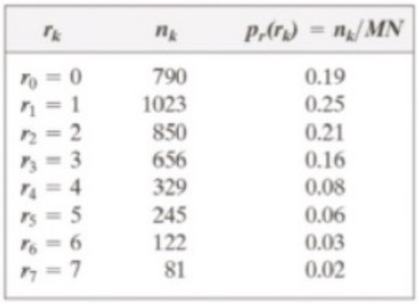


Figure 3

Figure 3 displays many photos both before and after equalisation. Additionally, it displays the histogram that was used to equalise the photos. In comparison to the original photos, the equalised images on the left of the figure are more detailed. This is so that the histogram can spread out the image's lower intensities. We can see that the last image, where gamma is 3, has a very low intensity because it is very dark. The high intensity is spread out by the histogram after equalisation has been applied.

Problem 4:



r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

TABLE 3.1 Intensity distribution and histogram values for a 3-bit, 64 × 64 digital image.

$$S_k = T(r_k) = (L-1)\sum_{j=0}^{k} P_r(r_j)$$

$$S_0 = 7 \cdot P_r(r_0) = 7(0.19) = 1.33$$

$$S_1 = 7 \cdot [P_r(r_0) + P_r(r_1)] = 7(0.19 + 0.25) = 3.08$$

$$S_2 = 7 \cdot [P_r(r_0) + P_r(r_1) + P_r(r_2)] = 7(0.19 + 0.25 + 0.21) = 4.55$$

$$S_3 = 7 \cdot [P_r(r_0) + P_r(r_1) + P_r(r_2) + P_r(r_3)] = 7(0.19 + 0.25 + 0.21 + 0.16) \simeq 5.67$$

$$S_4 = 7 \cdot [P_r(r_0) + P_r(r_1) + P_r(r_2) + P_r(r_3) + P_r(r_4)] = 7(0.19 + 0.25 + 0.21 + 0.16 + 0.08) = 6.23$$

$$S_5 = 7 \cdot [P_r(r_0) + P_r(r_1) + P_r(r_2) + P_r(r_3) + P_r(r_4) + P_r(r_5)]$$
$$= 7(0.19 + 0.25 + 0.21 + 0.16 + 0.08 + 0.06) = 6.65$$

$$S_6 = 7 \cdot [P_r(r_0) + P_r(r_1) + P_r(r_2) + P_r(r_3) + P_r(r_4) + P_r(r_5) + P_r(r_6)]$$
$$= 7(0.19 + 0.25 + 0.21 + 0.16 + 0.08 + 0.06 + 0.03) = 6.86$$

$$S_7 = 7 \cdot [P_r(r_0) + P_r(r_1) + P_r(r_2) + P_r(r_3) + P_r(r_4) + P_r(r_5) + P_r(r_6) + P_r(r_7)]$$
$$= 7(0.19 + 0.25 + 0.21 + 0.16 + 0.08 + 0.06 + 0.03 + 0.02) = 7$$

Equalized Histogram Values:

$S_0 = 1.33 \rightarrow 1$	$S_4 = 6.23 \rightarrow 6$
$S_1 = 3.08 \rightarrow 3$	$S_5 = 6.65 \rightarrow 7$
$S_2 = 4.55 \rightarrow 5$	$S_6 = 6.86 \rightarrow 7$
$S_3 = 5.67 \rightarrow 6$	$S_7 = 7 \rightarrow 7$

Probability Density Function (PDF) $(P_s(s_k))$

r_k	$P_r(r_k)$	S_k	$P_s(s_k)$	
$r_0=0$	0.19	0	0	
$r_1=1$	0.25	1	0.19	$\rightarrow$ 0.19
$r_2=2$	0.21	2	0	
$r_3=3$	0.16	3	0.25	$\rightarrow$ 0.25
$r_4=4$	0.08	4	0	
$r_5=5$	0.06	5	0.21	$\rightarrow$ 0.21
$r_6=6$	0.03	6	0.24	$\rightarrow$ 0.16 + 0.08
$r_7=7$	0.02	7	0.11	$\rightarrow$ 0.06 + 0.03 + 0.02

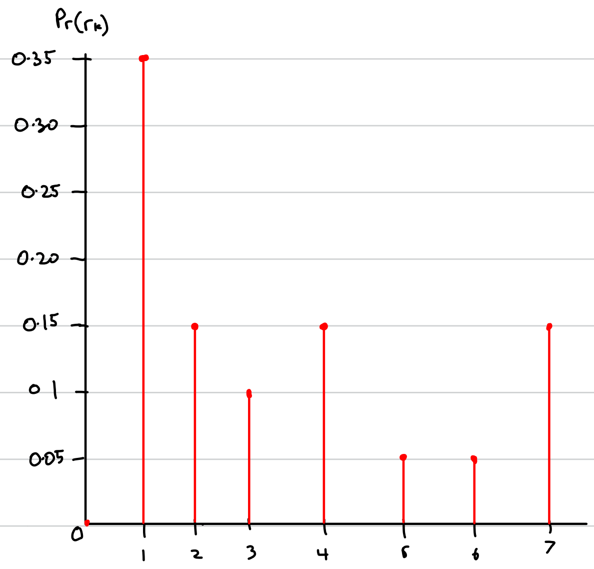
Problem 5

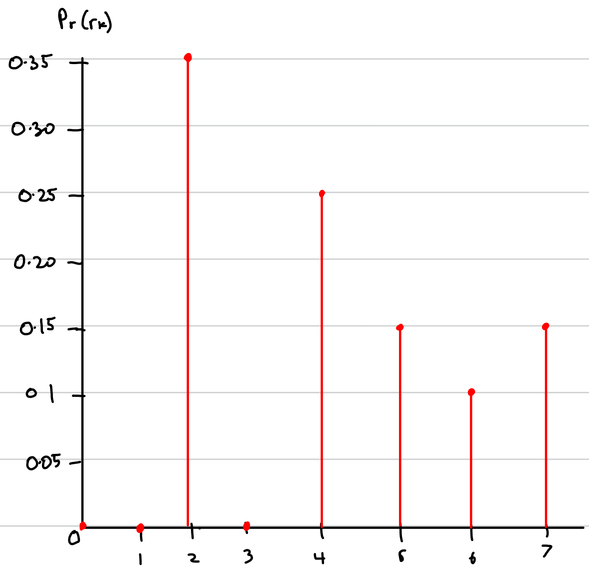
$M \times N = 5 \times 4 = 20$

$L =$ # of gray levels $= 8$

r_k	n_k	$P_r(r_k) = \frac{n_k}{MN} = \frac{n_k}{\sum n_k}$	$S_k = (L-1)\sum_{i=0}^{k} P_r(r_i)$	round (S_k)	$P_s(S_k)$
0	0	$0/20 = 0$	$7 \cdot 0/20 = 0$	0	0
1	7	$7/20 = 0.35$	$7 \cdot 7/20 = 2.45$	1	0
2	3	$3/20 = 0.15$	$7 \cdot 10/20 = 3.5$	2	0.35
3	2	$2/20 = 0.10$	$7 \cdot 12/20 = 4.2$	3	0
4	3	$3/20 = 0.15$	$7 \cdot 15/20 = 5.25$	4	0.25
5	1	$1/20 = 0.05$	$7 \cdot 16/20 = 5.6$	5	0.15
6	1	$1/20 = 0.05$	$7 \cdot 17/20 = 5.95$	6	0.10
7	3	$3/20 = 0.15$	$7 \cdot 20/20 = 7$	7	0.15

Histogram before equalization

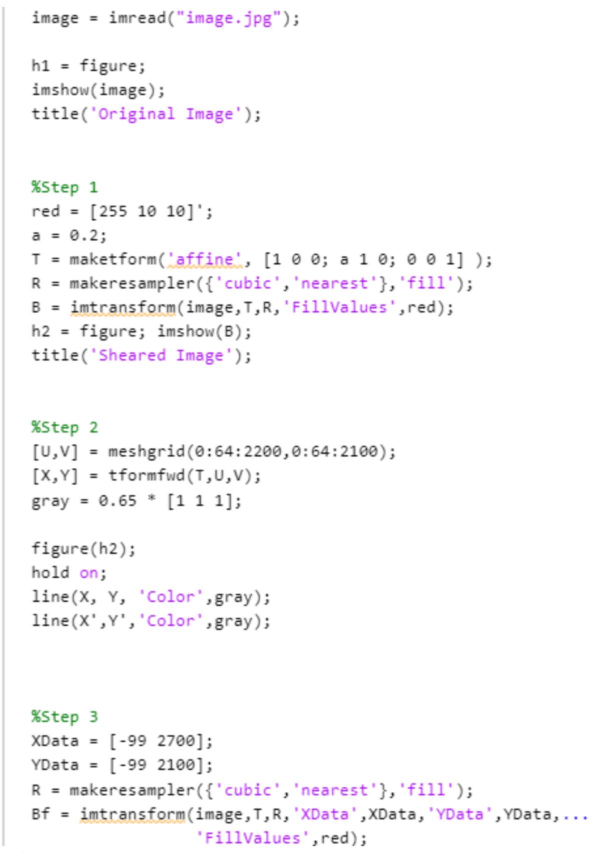




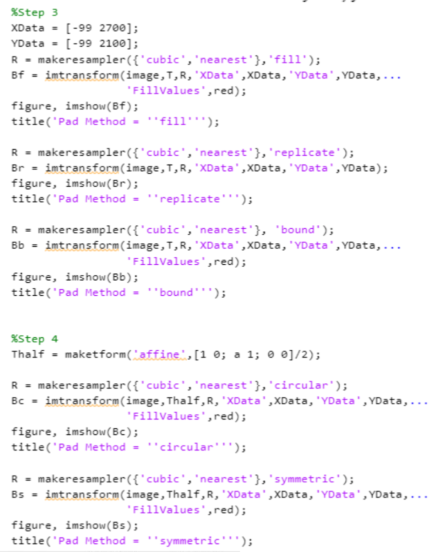
1	2	4	7	3
2	4	7	3	1
5	6	2	1	1
4	7	1	1	1

2	4	5	7	4
4	5	7	4	2
6	6	4	2	2
5	7	2	2	2

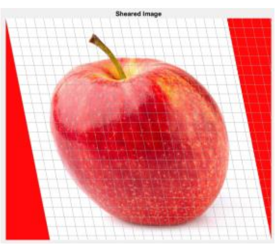
Part 2:



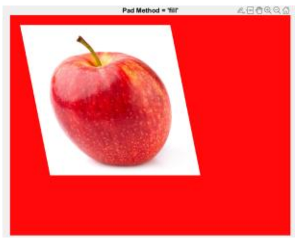
```
image = imread("image.jpg");

h1 = figure;
imshow(image);
title('Original Image');


%Step 1
red = [255 10 10]';
a = 0.2;
T = maketform('affine', [1 0 0; a 1 0; 0 0 1] );
R = makeresampler({'cubic','nearest'},'fill');
B = imtransform(image,T,R,'FillValues',red);
h2 = figure; imshow(B);
title('Sheared Image');


%Step 2
[U,V] = meshgrid(0:64:2200,0:64:2100);
[X,Y] = tformfwd(T,U,V);
gray = 0.65 * [1 1 1];

figure(h2);
hold on;
line(X, Y, 'Color',gray);
line(X',Y','Color',gray);



%Step 3
XData = [-99 2700];
YData = [-99 2100];
R = makeresampler({'cubic','nearest'},'fill');
Bf = imtransform(image,T,R,'XData',XData,'YData',YData,...
                 'FillValues',red);
```

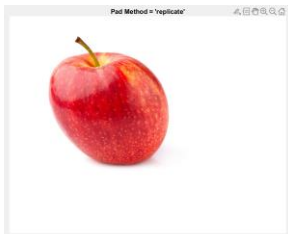


```
%Step 3
XData = [-99 2700];
YData = [-99 2100];
R = makeresampler({'cubic','nearest'},'fill');
Bf = imtransform(image,T,R,'XData',XData,'YData',YData,...
                 'FillValues',red);
figure, imshow(Bf);
title('Pad Method = ''fill''');

R = makeresampler({'cubic','nearest'},'replicate');
Br = imtransform(image,T,R,'XData',XData,'YData',YData);
figure, imshow(Br);
title('Pad Method = ''replicate''');

R = makeresampler({'cubic','nearest'}, 'bound');
Bb = imtransform(image,T,R,'XData',XData,'YData',YData,...
                 'FillValues',red);
figure, imshow(Bb);
title('Pad Method = ''bound''');


%Step 4
Thalf = maketform('affine',[1 0; a 1; 0 0]/2);

R = makeresampler({'cubic','nearest'},'circular');
Bc = imtransform(image,Thalf,R,'XData',XData,'YData',YData,...
                 'FillValues',red);
figure, imshow(Bc);
title('Pad Method = ''circular''');

R = makeresampler({'cubic','nearest'},'symmetric');
Bs = imtransform(image,Thalf,R,'XData',XData,'YData',YData,...
                 'FillValues',red);
figure, imshow(Bs);
title('Pad Method = ''symmetric''');
```

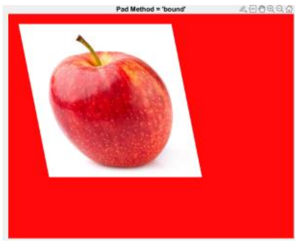


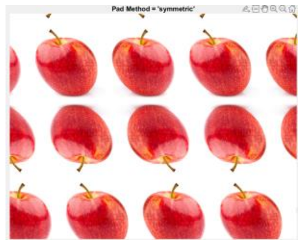
Original Image *Sheared Image*



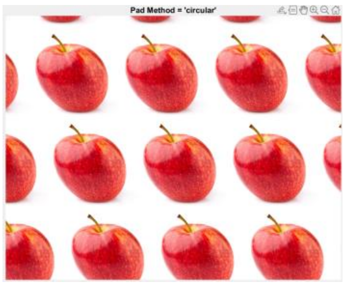


Pad Method = 'fill' *Pad Method = 'replicate'*





Pad Method = 'bound' *Pad Method = 'symmetric'*



Pad Method = 'circular'

The image was subjected to a straightforward shear transformation in step 1. An affine transformation that shifts a point of the picture away from the axis is known as a simple shear transformation. It becomes more bent the farther it is from the other axis. Maketform uses the simple shear matrix values to create an affine. This can be seen in the photo with the description "sheared image." If we compare it to the original photo, we can see how it was altered.

In step 2, we were instructed to draw lines so that we could observe how the lines were affected by the basic shear transformation and how they had been bent. The sheared image shows this as well.

We examine the various cushioning possibilities in step 3 of the process. The first one is fill, where we merely apply red colour to the image padding. The graphic with the caption "Pad Method = 'fill'" illustrates this. The duplicate padding method is the next padding technique we employ. The replicate padding method duplicates the source image, flips it, and then uses that as the new padding for the image. This is demonstrated in the image "Pad Method ='replicate'"; however, because of the white background of the example image, it is not quite evident. The fill padding technique is quite similar to bound padding. In contrast to the fill method, the bound padding method clearly delineates the borders between the padding and the input image. The graphic shows this as Pad Method = "bound."

In the final phase, we test out two additional padding techniques: symmetric padding and circular padding. Figures "Pad Method = 'circular'" and "Pad Method ='symmetric'" provide illustrations of these. The symmetric padding method uses the input image's mirror to fill the padding instead of using copies of the input image like the circular padding method does.