NOTE: some of these operations only work with Gray images.

1. Split an RGB image into its three channels.

```
% this code produces three grayscale images.
tmp = imread( 'PICTURE.ext');

red = tmp(:,:,1);
green = tmp(:,:,2);
blue = tmp(:,:,3);

figure,imshow(red),figure,imshow(blue),figure,imshow(green);
```

2. Split an RGB image into its three channels (2).

```
% this code produces three images & each image has only one color in it.
img = imread('PICTURE.ext');
[row,col,del] = size(img); %del is a unused but important variable

imred = img;
imblue = img;
imgreen = img;
for i =1:row
    for j = 1:(col)
        imred(i,j,2) = 0;
        imred(i,j,3) = 0;

        imgreen (i,j,1) = 0;
        imgreen (i,j,3) = 0;

        imblue (i,j,1) = 0;
        imblue (i,j,2) = 0;
    end
end

figure,imshow(imred),figure,imshow(imblue),figure,imshow(imgreen);
```

3. Add/sub value to brighten/darken an image.

```
img = imread('PICTURE.ext');

[row,col,tt] = size(img);

for i=1:row
    for j=1:col
        img(i,j,1) = img(i,j,1)+100; %this number can change based on the question
        img(i,j,2) = img(i,j,2)+100;
        img(i,j,3) = img(i,j,3)+100;
    end
end

imshow('PICTURE.ext'),figure,imshow(img);
```

4. Add/sub value to brighten/darken an image (2).

```
img = imread('PICTURE.ext');

[row,col,tt] = size(img);

img =img +100;

imshow('PICTURE.ext'),figure,imshow(img);
```

5. Color to gray (using average method).

```
% these three lines extract the color channels.
img = imread('PICTURE.ext');
R=img(:,:,1);
G=img(:,:,2);
B=img(:,:,3);

% this is the average weighted method.
% it takes every channel and multiplies it by 1/3.

% can be replaced by: gray_img = (R + G + B) / 3;
gray_img=.333*R + .333*G + .333*B;
gray_img = uint8(gray_img);

imshow(img),figure,imshow(gray_img);
```

6. Color to gray (using lightness method).

```
img = imread('PICTURE.ext');

gray_img = ( max(img,[],3) + min(img,[],3) ) / 2;

imshow('PICTURE.ext'),figure,imshow(gray_img);
```

7. color to gray (using weighted method).

```
% Read in a color image
img = imread('PICTURE.ext');

% weights of each color
R = img(:,:,1);
G = img(:,:,2);
B = img(:,:,3);

gray_img = (0.299 * R) + (0.587 * G) + (0.114 * B);

imshow('PICTURE.ext'),figure,imshow(gray_img);
```

8. Thresholding: gray to black & white.

```
img = imread('PICTURE.ext');
thresh_val = 50;

% this sets any pixel higher than the thresh value to true (white)
% and any pixel value thats lower gets set to false (black)
thresh_img = img > thresh_val;
out_img = uint8(thresh_img) * 255;

imshow('PICTURE.ext'),figure,imshow(thresh_img);
```

9. Thresholding: gray to black & white (2).

```
img = imread('PICTURE.ext');
thresh_val = 50;
[M, N] = size(img);
thresh_img = zeros(M, N);
for i = 1:M
    for j = 1:N
        if img(i,j) > thresh_val
            thresh_img(i,j) = 1;
        else
            thresh_img(i,j) = 0;
        end
    end
end
```

10. Transpose: rotate 90 degrees.

```
img = imread('PICTURE.ext');
[row,col] = size(img);
transposed_img = uint8(zeros(col,row));

for i = 1:row
    for j = 1:col
        transposed_img(j,i) = img(i,j);
    end
end
imshow('PICTURE.ext'),figure,imshow(transposed_img);
```

11. Flip image.

```
img = imread('PICTURE.ext');
[row,col] = size(img);
flipped = zeros(100,100);
for i = 1:row
    for j = 1:col
        flipped(i,col+1-j) = img(i,j);
    end
end
flipped = uint8(flipped);
imshow('PICTURE.ext'),figure,imshow(flipped);
```

12. Negative image.

```
img = imread('PICTURE.ext');
[row,col] = size(img);
negated = zeros(100,100);
for i = 1:row
    for j = 1:col
        negated(i,j) = 255-img(i,j);
    end
end
negated = uint8(negated);
imshow('PICTURE.ext'),figure,imshow(negated);
```

13. Zoom image (using pixel replication).

```
img =imread('PICTURE.ext');
zoom_factor=2;
[m,n]=size(img);
zoomed_img = zeros(zoom_factor*size(img),class(img));

for i=1:m
  for j=1:n
      for k=1:zoom_factor
        for t=1:zoom_factor
              zoomed_img((i-1)*zoom_factor+t, (j-1)*zoom_factor+k)=img(i,j);
         end
      end
  end
end
imshow(img),figure,imshow(zoomed_img);
```

14. Zoom image (using zero order hold) [Extra??].

```
img = imread('PICTURE.ext');
zoom_factor = 2;

% Calculate the new image size after zooming
[m,n] = size(img);
new_m = m*zoom_factor;
new_n = n*zoom_factor;

zoomed_img = zeros(new_m,new_n,'uint8');

for i = 1:new_m
    for j = 1:new_n
        x = ceil(i/zoom_factor);
        y = ceil(j/zoom_factor);
        zoomed_img(i,j) = img(x,y);
    end
end
imshow(img), figure, imshow(zoomed_img);
```

15. zoom image (using k-times) [Extra??].

```
img = imread('input_image.png');
k = 2;

[m,n] = size(img);
new_m = k*m;
new_n = k*n;

zoomed_img = zeros(new_m,new_n,'uint8');

for i = 1:m
    for j = 1:n
        x = (i-1)*k+1;
        y = (j-1)*k+1;
        zoomed_img(x:x+k-1, y:y+k-1) = img(i,j);
    end
end
imshow(img), figure, imshow(zoomed_img);
```

16. segment a particular range from an image (set rest to zero).

```
img = imread('PICTURE.ext');

segmented_img=img;
[row,col,temp] =size(img);

lower = 50; % both upper & lower bounds can change
upper =120;
for i =1:row
    for j=1:col
        if(img(i,j)<lower || img(i,j)>upper)
            segmented_img(i,j) = 0;
        end
    end
end
imshow(img), figure, imshow(segmented_img);
```

17. Segment a particular range from an image (set range to highest value).

```
img = imread('PICTURE.ext');

segmented_img=img;
[row,col,temp] =size(img);

lower= 50; % both upper & lower bounds can change
upper =120;
for i =1:row
    for j=1:col
        if(img(i,j)>lower && img(i,j)<upper)
            segmented_img(i,j) = 255;
        end
    end
end
imshow(img), figure, imshow(segmented_img);
```

18. Contrast stretching (setting new max & min values).

```
image = imread('PICTURE.ext');
image = im2double(image);

min_val = min(image(:));
max_val = max(image(:));

% these two variables are the new Min & Max
desired_min = 0;
desired_max = 1;

adjusted_image = (image - min_val) * ((desired_max - desired_min) / (max_val - min_val)) +
desired_min;

adjusted_image = uint8(adjusted_image * 255);

imshow(image);
figure,imshow(adjusted_image);
```

19. Contrast optimization (normalization).

```
img=imread('PICTURE.ext');
conimage=(img / 250-0)*255;

imshow(img);
figure, imshow(conimage);
```

20. Creating histogram manually (frequency).

```
img = imread('PICTURE.ext');
[row,col] = size(img);
hist = zeros(1,256);
for i= 1:row
    for j=1:col
        for pixel_val =1:256
            if img(i,j) == pixel_val -1
                hist(pixel_val) = hist(pixel_val)+1;
               % you can start from 0 to 255 but change every (pixel_val) to (pixel_val-1)
            end
        end
    end
end

figure,imhist(img),figure,plot(hist),figure,bar(hist),figure,stem(hist);
```

21. Manual histogram equalization.

```
% assuming variable (hist) exists & [row, col] also exist + img (use operation 20)

% finding the cumualtive sum of all values.
 num_pixels = row*col;

 cumulative(1)=hist(1)/num_pixels;
 for i=2:256 % starting from the second cumulative value to 256
     cumulative(i) = cumulative(i-1) + (hist(i)/num_pixels);
 end
 cumulative(:) = cumulative(:)*255;

 for i=1:row
     for j=1:col
         equalized_img(i,j) = cumulative(equalized_img(i,j)+1);
     end
 end

 equalized_hist = zeros(1,256);
 for i= 1:row
    for j=1:col
        for pix_val =1:256
            if img(i,j) == pix_val -1
                equalized_hist(pix_val) = equalized_hist(pix_val)+1;
               %%you can start from 0 to 255 but change every (r) to (r-1)
            end
        end
    end
 end
 subplot(2,2,1), imshow(img);
 subplot(2,2,2), plot(hist);
 subplot(2,2,3), imshow(equalized_img);
 subplot(2,2,4), plot(equalized_hist);
```

22. Cut a piece of an image.

```
img = imread('PICTURE.ext');
[rows,cols] = size(img);

removeX = 0;
removeY = 100;
im2=img(1:cols-removeX,1:rows-removeY);
imshow(img), figure, imshow(im2);
```

23. Apply mean filter 7*7(blur image).

```
img = imread('PICTURE.ext');

% neighborhood size must be odd (must have a center pixel).
kernel = ones(7, 7) / (7 * 7);

filtered_img = conv2(double(img), kernel, 'same');
filtered_img = uint8(filtered_img);

imshowpair(img,filtered_img,'montage');
```

24. Simpler histogram equalization (using built-in functions).

```
img = imread('PICTURE.ext');
histogram = imhist(img);

% Calculate the cumulative distribution function (CDF)
cdf = cumsum(histogram);

num_pixels = numel(img);

% Normalize the CDF
cdf_normalized = cdf / num_pixels;

adjusted_img = cdf_normalized(img+1);
adjusted_img = uint8(adjusted_img * 255);

imshowpair(img,adjusted_img,'montage');
```

25. Mean filter 3*3 (no pre-built functions).

```
img = imread('PICTURE.ext');   [row, col] = size(img);

x=im2double(img);   blurred_img = zeros(row,col,'double');

filter = ones(3,3)/(3*3);   [filter_row, filter_col] = size(filter);

ficenter = floor(filter_row/2)+1;   fjcenter = floor (filter_col/2)+1;

for i=ficenter:row-ficenter+1
    for j=fjcenter:col-fjcenter+1
        valueij = 0;
        for k=i-1:i+1
            for l= j-1:j+1
               valueij = valueij+double(x(k,l));
            end
        end
        blurred_img(i,j) = double(valueij/9);
    end
end
imshowpair(img,blurred_img,'montage');
```

26. Hide a message in an image.

```
img = imread('PICTURE.ext');
message = input('enter your message: ','s');
len = length(message)*8;

% finding the ascii value of each letter
ascii_value = uint8(message);

binary_message = transpose(dec2bin(ascii_value,8));

% representing all bits in a 1D array (column by column)
binary_message = binary_message(:);

bin_num_message = str2num(binary_message);

[row,col] = size(img);
embedded_img = img;

embed_counter = 1;
for i = 1:row
    for j = 1:col
        if(embed_counter<=len)
             % same as using var%2 in other languages (finding if the numebr
             % is even || odd to find if the last bit is a 1 || 0).
            LSB = mod(double(img(i,j)),2);
            temp = double(xor(LSB,bin_num_message(embed_counter)));

            embedded_img(i,j) = img(i,j)+temp;
            embed_counter = embed_counter + 1;
        end
    end
end
imshow(embedded_img);
```

27. Extract a hidden message.

```
embedded_img = imread('PICTURE.ext') % load the embedded img
counter =1;
[row,col] = size(embedded_img);
for i = 1:row
    for j = 1:col
        if(counter<=len)
            extracted_bits(counter,1) = mod(double(embedded_img(i,j)),2);
            counter = counter + 1;
        end
    end
end

binvalues = [ 128 64 32 16 8 4 2 1];

% changing the 1D array to a matrix
binMatrix = reshape(extracted_bits,8,(len/8));
textString = char(binvalues*binMatrix);

disp(textString);
```

28.