1.1(a)  
 x = randperm(50);  
  
 x has random integers generated from 1 to 50  
  
1.2(b)  
 a = [1:2:10; 11:2:20]';

b = a(2,:);  
  
 a is a 2x5 matrix where the first row contains odd numbers starting from 1 to 10 (1,3,5,7,9), and the second row contains odd numbers from 11 to 20  
 b is a 1x5 matrix which has only the second row of the a array.  
  
1.3(c)  
 f = randn(10,2);  
 g = f(find(f < 1));  
  
 f is 10x2 matrix of random numbers from a standart normal distribution  
 g matrix has the values of f that are less than 1  
  
1.4(d)  
 x = 0.5 .\* ones(1,10);  
 y = 0.5 + zeros(1,length(x));  
 z = x + y;  
   
 x is a vector of length 10 filled with the value of 0.5  
 y is a vector of length 10 filled with the value of 0.5  
 z is a vector of length 10 where each element is 1  
   
1.5(e)  
 a = [1:100];  
 b = a([end:-1:1]);  
  
 a is a vector containing numbers from 1 to 100.   
 a([end:-1:1]) creates a vector b by indexing a starting from the last element of a then going back to 1 until first element, so in a reverse order.  
 b is a vector containing numbers from 100 to 1 in reverse order  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
2.1(a)

A\_1D = reshape(A, 1, []);

sorted\_intensities = sort(A\_1D, 'descend');

% Plot the descended intensities

plot(sorted\_intensities);

title('Sorted Intensities in Decreasing Order');  
 xlabel('Pixel Index');  
 ylabel('Intensity Value');

2.2(b)

threshold = 150;

RGB = zeros([size(A), 3]);

RGB(:,:,3) = (A > threshold) \* 255;

% Display the RGB image

imshow(RGB);  
 title('Color Image with Blue Values for Intensities > t');

2.3(c)  
 X = A(1:25, 1:25);

2.4(d)  
 mean\_intensity = mean(A(:))  
 image\_eachpixel\_subtracted\_by\_mean = A – mean\_intensity;  
 imshow(image\_eachpixel\_subtracted\_by\_mean, []);  
2.5(e)  
 y\_2 = [1:12]  
 z\_2 = reshape(y, 4, 3);   
2.6(f)  
 v = [1 8 8 2 1 3 9 8]  
 occurrences\_of\_8 = sum(v == 8);

3.1(a)

function flippedImage = flipVertical(image)

% Get image dimensions

[rows, cols] = size(image);

% Allocate memory for flipped image (same size and type as original)

flippedImage = zeros(rows, cols, class(image));

% Loop through each pixel and reverse order

for row = 1:rows

for col = 1:cols

flippedImage(rows - row + 1, col) = image(row, col);

end

end

end

3.2(b)

function flippedImage = flipHorizontal(image)

% Get image dimensions

[rows, cols] = size(image);

% Allocate memory for flipped image

flippedImage = zeros(rows, cols, class(image));

% Loop through each pixel and reverse order

for row = 1:rows

for col = 1:cols

flippedImage(row, cols - col + 1) = image(row, col);

end

end

end

3.3(c)

function negativeImage = negativeImage(image)

% Check if grayscale (avoid unnecessary conversion)

if isa(image, 'uint8')

maxValue = 255;

else

maxValue = 1; % Assuming double precision for grayscale

end

% Create negative image by subtracting from maximum value

negativeImage = maxValue - image;

end

3.4(d)

function swappedImage = swapRG(image)

% Check for RGB image (avoid errors)

if ~isa(image, 'uint8') || size(image, 3) ~= 3

error('Function only works for uint8 RGB images');

end

% Swap red and green channels using temporary variable

temp = image(:, :, 1);

image(:, :, 1) = image(:, :, 2);

image(:, :, 2) = temp;

% Assign swapped image (avoid unnecessary copy)

swappedImage = image;

end

3.5(e)

function averagedImage = averageMirror(image)

% Flip horizontally to create mirror image

mirrorImage = flipHorizontal(image);

% Cast to double for element-wise operations (avoid overflow)

imageDouble = double(image);

mirrorDouble = double(mirrorImage);

% Average pixel values (round to avoid potential precision issues)

averagedImage = uint8(round((imageDouble + mirrorDouble) / 2));

end

3.6(f)

function modifiedImage = randomModify(image)

% Check for grayscale image

if ~isa(image, 'uint8')

error('Function only works for uint8 grayscale images');

end

% Generate random value between 0 and 255

randomValue = uint8(randi(255));

% Decide on addition or subtraction (randomly)

if randi(2) == 1

modifiedImage = image + randomValue;

else

modifiedImage = image - randomValue;

end

% Clip values to 0-255

modifiedImage = min(max(modifiedImage, 0), 255);

end

script

image = imread('/Users/kemalyagizozcan/Desktop/leaf.jpg');

% Check if color or grayscale (convert to grayscale if color)

if size(image, 3) == 3

image = rgb2gray(image);

end

% Apply transformations and store results

flippedV = flipVertical(image);

flippedH = flipHorizontal(image);

negative = negativeImage(image);

% Handle color image check for swapRG (replace if not using RGB image)

if size(image, 3) == 3

swapped = swapRG(image);

else

swapped = image; % Avoid errors if not RGB

end

averaged = averageMirror(image);

modified = randomModify(image);

% Create figure and subplots

figure;