Image Processing – Lab 01

Aim: Explore the Octave GUI and matrix related operation, image related different function.

Description:

- For matrix creation :

```
A = [6 43 2 11 87;
     12 6 34 0 5;
     34 18 7 41 9]
```

Above code creates 3*5 matrix A.

<u>Different operation on Image:</u>

- imread ("filename"):

  Read an image as a matrix from the file name or from the online resource url.

- imwrite (img,"filename"):

  Write images in various file formats.

  The image img can be binary, grayscale, RGB, or multi-dimensional image.

- rgb2gray (rgb_img):

  Transform an image or colormap from red-green-blue (RGB) color space to a grayscale intensity image. The input may be of class uint8, uint16, int8, int16, single or double. The output is of the same class as the input.

- im2bw (img):

  Convert image to binary, black and white, by threshold. The input image img can be either be a grayscale or RGB image.

- imshow(img):

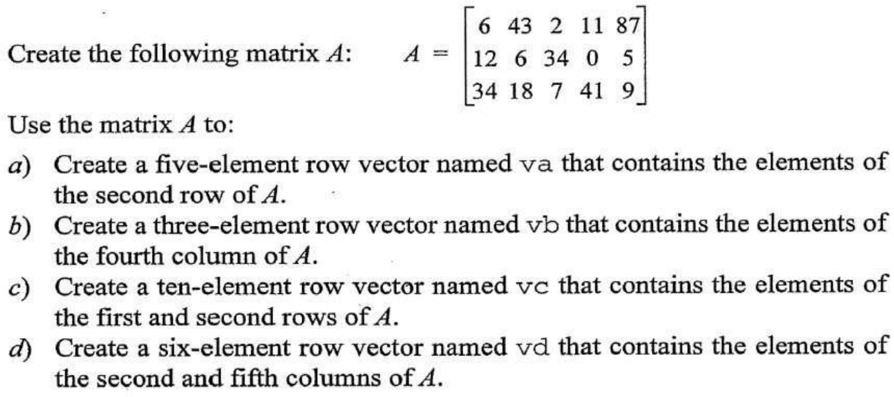
  Display the imge img, where img can be a 2-dimensional (grayscale image) or a 3-dimensional (RGB) matrix.

- imresize(im,scale)

  Resize image with interpolation. Scales image im by a factor scale or into the size M rows by N columns.

Assignment

1.



Create the following matrix A: $A = \begin{bmatrix} 6 & 43 & 2 & 11 & 87 \\ 12 & 6 & 34 & 0 & 5 \\ 34 & 18 & 7 & 41 & 9 \end{bmatrix}$

Use the matrix A to:

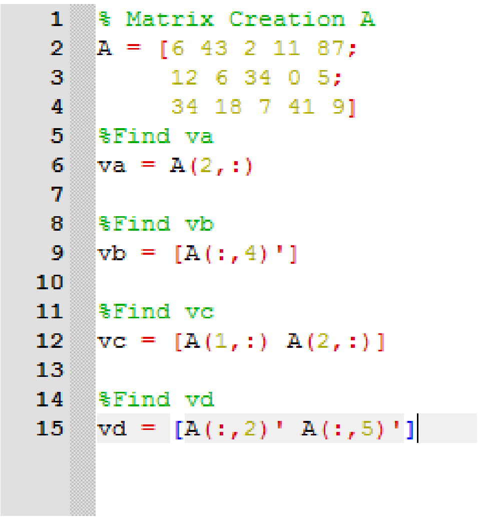
a) Create a five-element row vector named va that contains the elements of the second row of A.

b) Create a three-element row vector named vb that contains the elements of the fourth column of A.

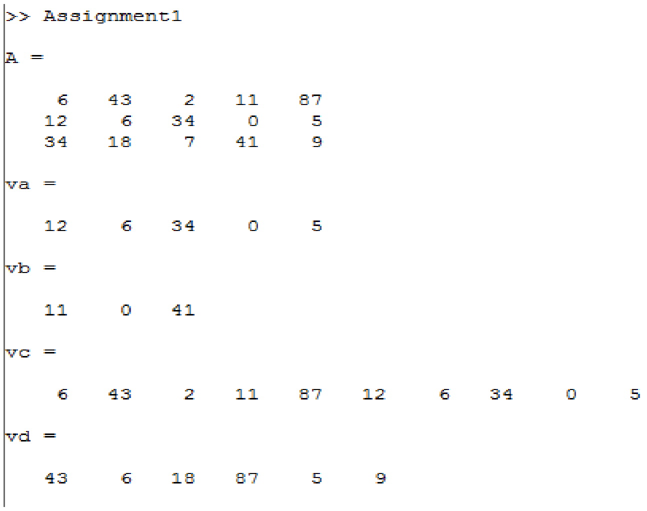
c) Create a ten-element row vector named vc that contains the elements of the first and second rows of A.

d) Create a six-element row vector named vd that contains the elements of the second and fifth columns of A.

Code & Output:



```
% Matrix Creation A
A = [6 43 2 11 87;
     12 6 34 0 5;
     34 18 7 41 9]
%Find va
va = A(2,:)

%Find vb
vb = [A(:,4)']

%Find vc
vc = [A(1,:) A(2,:)]

%Find vd
vd = [A(:,2)' A(:,5)']
```



```
>> Assignment1

A =

     6    43     2    11    87
    12     6    34     0     5
    34    18     7    41     9

va =

    12     6    34     0     5

vb =

    11     0    41

vc =

     6    43     2    11    87    12     6    34     0     5

vd =

    43     6    18    87     5     9
```

2.

Create the following three matrices:

$$A = \begin{bmatrix} 5 & 2 & 4 \\ 1 & 7 & -3 \\ 6 & -10 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 11 & 5 & -3 \\ 0 & -12 & 4 \\ 2 & 6 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 7 & 14 & 1 \\ 10 & 3 & -2 \\ 8 & -5 & 9 \end{bmatrix}$$

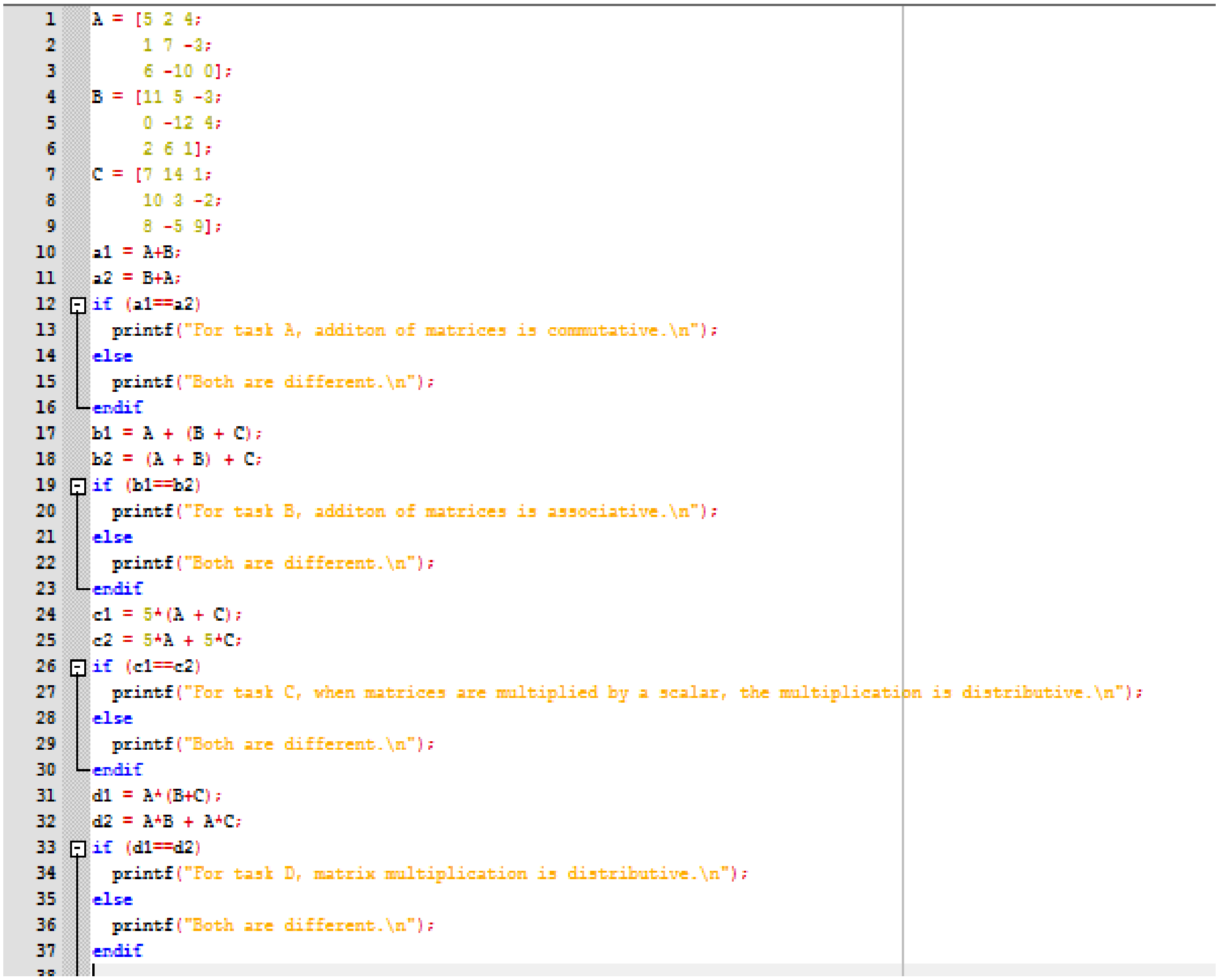
a) Calculate $A + B$ and $B + A$ to show that addition of matrices is commutative.

b) Calculate $A + (B + C)$ and $(A + B) + C$ to show that addition of matrices is associative.

c) Calculate $5(A + C)$ and $5A + 5C$ to show that, when matrices are multiplied by a scalar, the multiplication is distributive.

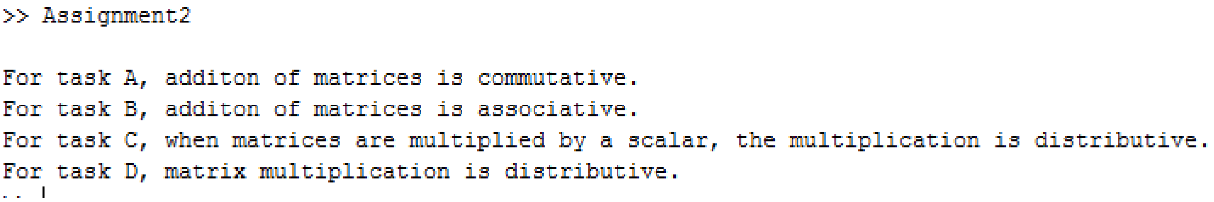
d) Calculate $A*(B + C)$ and $A*B + A*C$ to show that matrix multiplication is distributive.

Code :



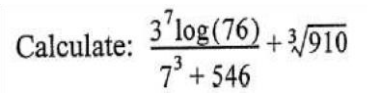
```
A = [5 2 4;
     1 7 -3;
     6 -10 0];
B = [11 5 -3;
     0 -12 4;
     2 6 1];
C = [7 14 1;
     10 3 -2;
     8 -5 9];
a1 = A+B;
a2 = B+A;
if (a1==a2)
  printf("For task A, additon of matrices is commutative.\n");
else
  printf("Both are different.\n");
endif
b1 = A + (B + C);
b2 = (A + B) + C;
if (b1==b2)
  printf("For task B, additon of matrices is associative.\n");
else
  printf("Both are different.\n");
endif
c1 = 5*(A + C);
c2 = 5*A + 5*C;
if (c1==c2)
  printf("For task C, when matrices are multiplied by a scalar, the multiplication is distributive.\n");
else
  printf("Both are different.\n");
endif
d1 = A*(B+C);
d2 = A*B + A*C;
if (d1==d2)
  printf("For task D, matrix multiplication is distributive.\n");
else
  printf("Both are different.\n");
endif
```

Output:



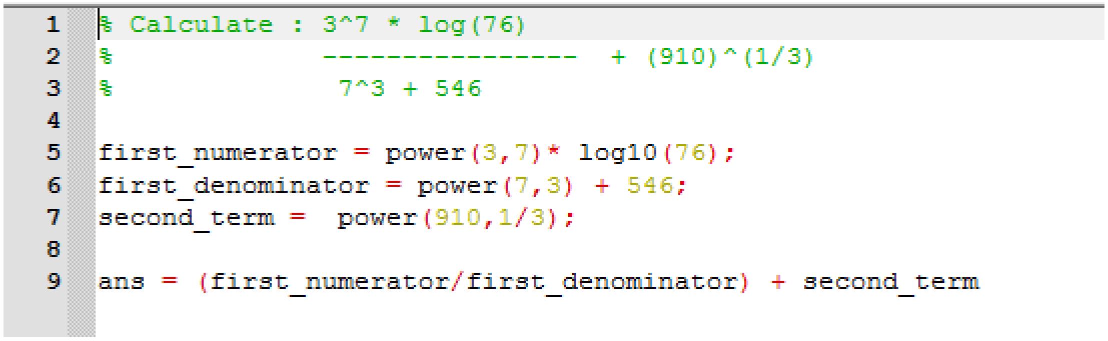
```
>> Assignment2

For task A, additon of matrices is commutative.
For task B, additon of matrices is associative.
For task C, when matrices are multiplied by a scalar, the multiplication is distributive.
For task D, matrix multiplication is distributive.
```

3.



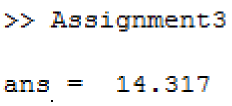
Calculate: $\frac{3^7\log(76)}{7^3+546}+\sqrt[3]{910}$

Code :



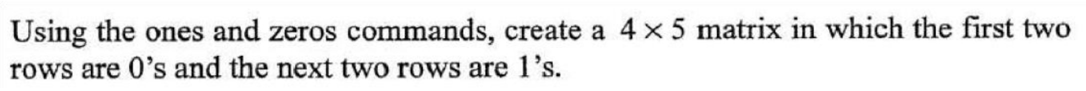
```
% Calculate : 3^7 * log(76)
%             ----------------  + (910)^(1/3)
%              7^3 + 546

first_numerator = power(3,7)* log10(76);
first_denominator = power(7,3) + 546;
second_term =  power(910,1/3);

ans = (first_numerator/first_denominator) + second_term
```

Output :



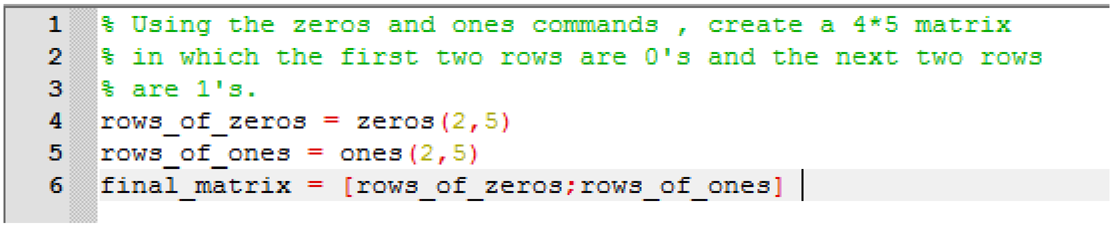
```
>> Assignment3

ans =  14.317
```

4.



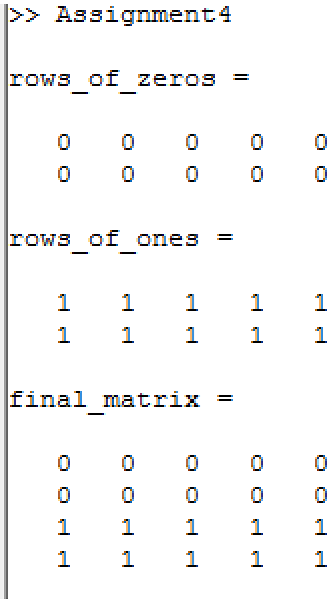
Using the ones and zeros commands, create a 4×5 matrix in which the first two rows are 0’s and the next two rows are 1’s.

Code :



```
% Using the zeros and ones commands , create a 4*5 matrix
% in which the first two rows are 0's and the next two rows
% are 1's.
rows_of_zeros = zeros(2,5)
rows_of_ones = ones(2,5)
final_matrix = [rows_of_zeros;rows_of_ones]
```

Output :



```
>> Assignment4

rows_of_zeros =

     0     0     0     0     0
     0     0     0     0     0

rows_of_ones =

     1     1     1     1     1
     1     1     1     1     1

final_matrix =

     0     0     0     0     0
     0     0     0     0     0
     1     1     1     1     1
     1     1     1     1     1
```

5. Take your own photo(RGB image) and create following images and save them for future use.

1) Gray scale image

2) Black and white image

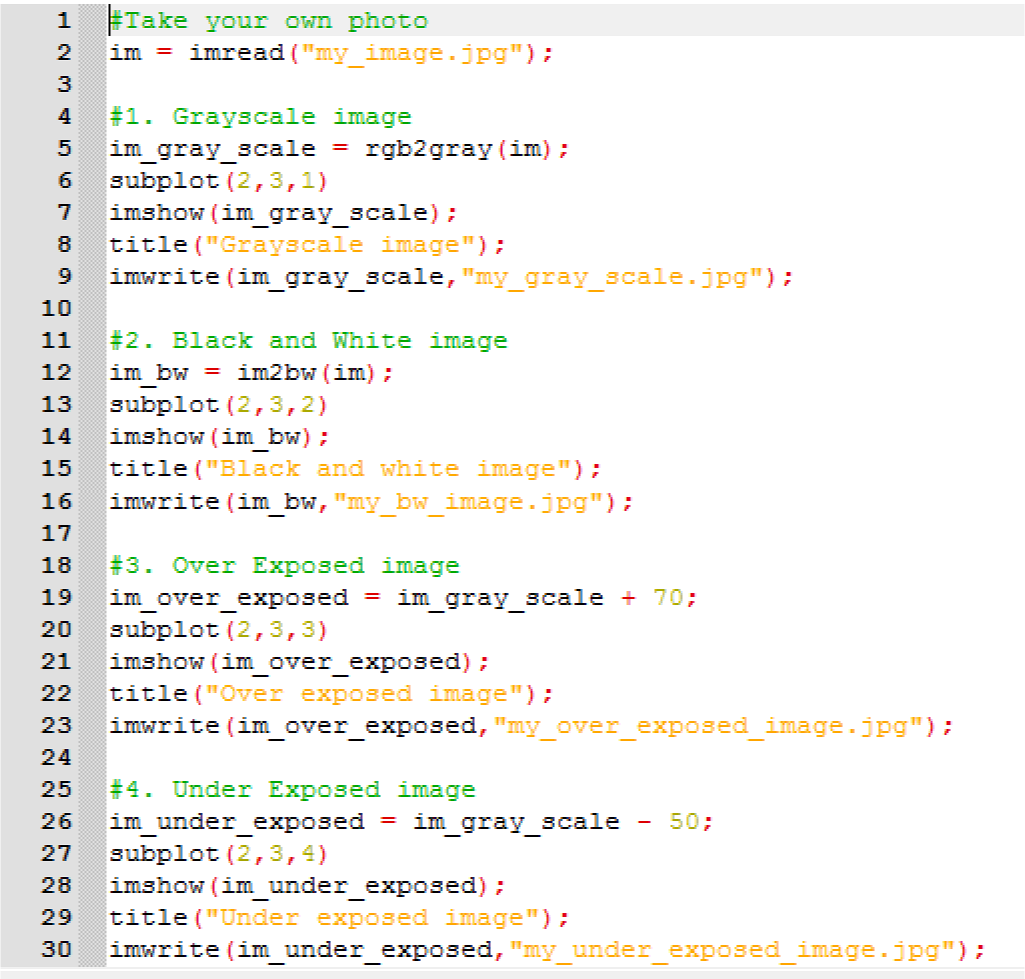
3) Over Exposed image

4) Under Exposed image

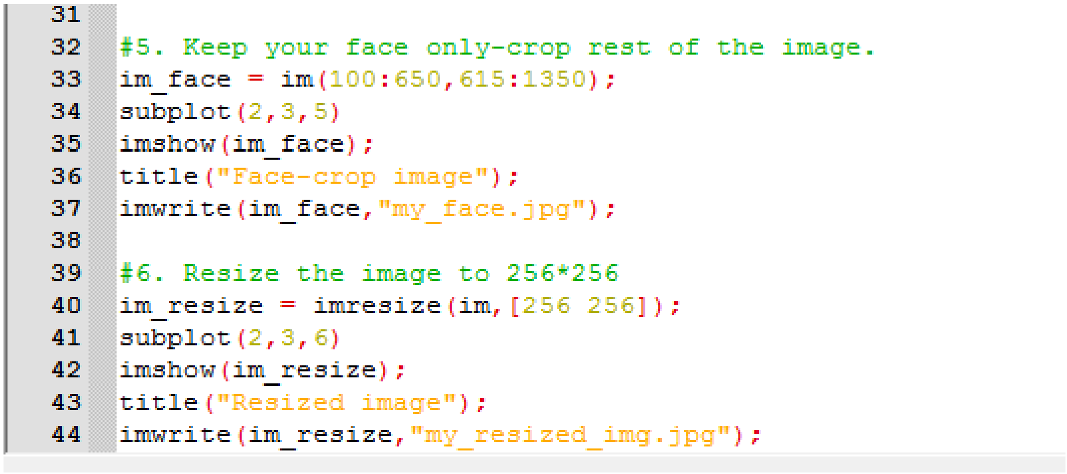
5).keep your face only-crop rest of the image.

6).Resize the image to 256*256.

Code :

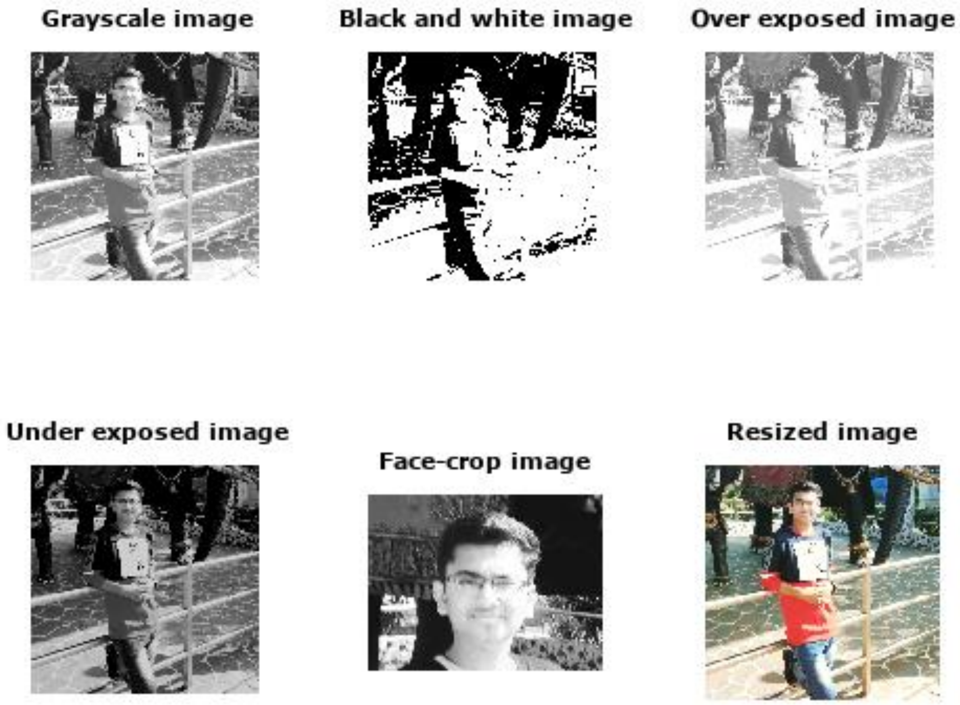


```
#Take your own photo
im = imread("my_image.jpg");

#1. Grayscale image
im_gray_scale = rgb2gray(im);
subplot(2,3,1)
imshow(im_gray_scale);
title("Grayscale image");
imwrite(im_gray_scale,"my_gray_scale.jpg");

#2. Black and White image
im_bw = im2bw(im);
subplot(2,3,2)
imshow(im_bw);
title("Black and white image");
imwrite(im_bw,"my_bw_image.jpg");

#3. Over Exposed image
im_over_exposed = im_gray_scale + 70;
subplot(2,3,3)
imshow(im_over_exposed);
title("Over exposed image");
imwrite(im_over_exposed,"my_over_exposed_image.jpg");

#4. Under Exposed image
im_under_exposed = im_gray_scale - 50;
subplot(2,3,4)
imshow(im_under_exposed);
title("Under exposed image");
imwrite(im_under_exposed,"my_under_exposed_image.jpg");
```



```

#5. Keep your face only-crop rest of the image.
im_face = im(100:650,615:1350);
subplot(2,3,5)
imshow(im_face);
title("Face-crop image");
imwrite(im_face,"my_face.jpg");

#6. Resize the image to 256*256
im_resize = imresize(im,[256 256]);
subplot(2,3,6)
imshow(im_resize);
title("Resized image");
imwrite(im_resize,"my_resized_img.jpg");
```

Output :



6. Take your own photo and process them for following results using loop controling structures.

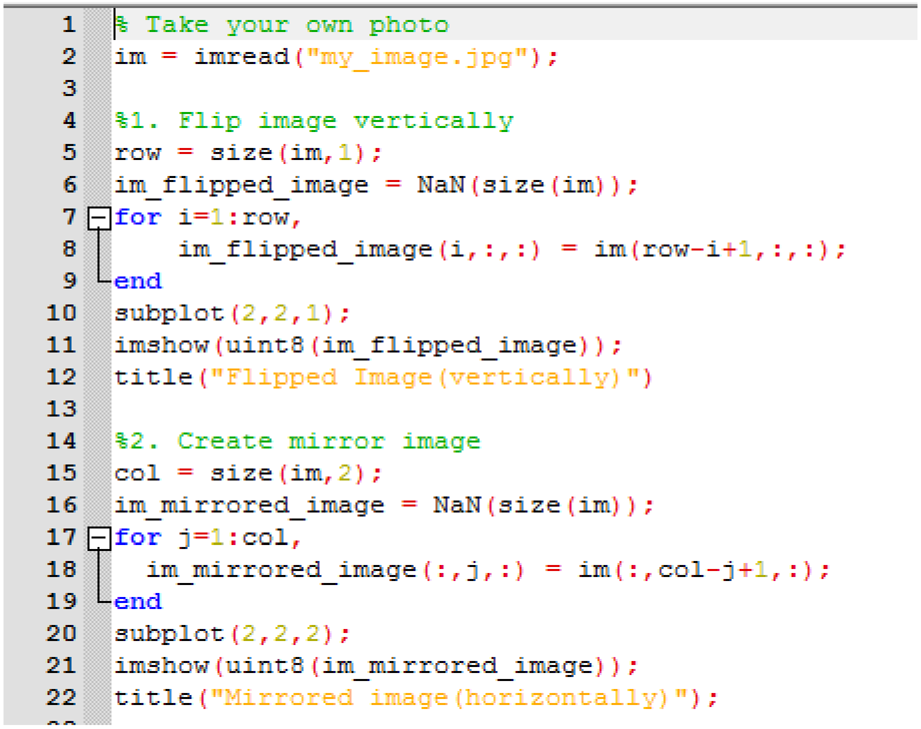
1) flip your image vertically

2) create the mirror image

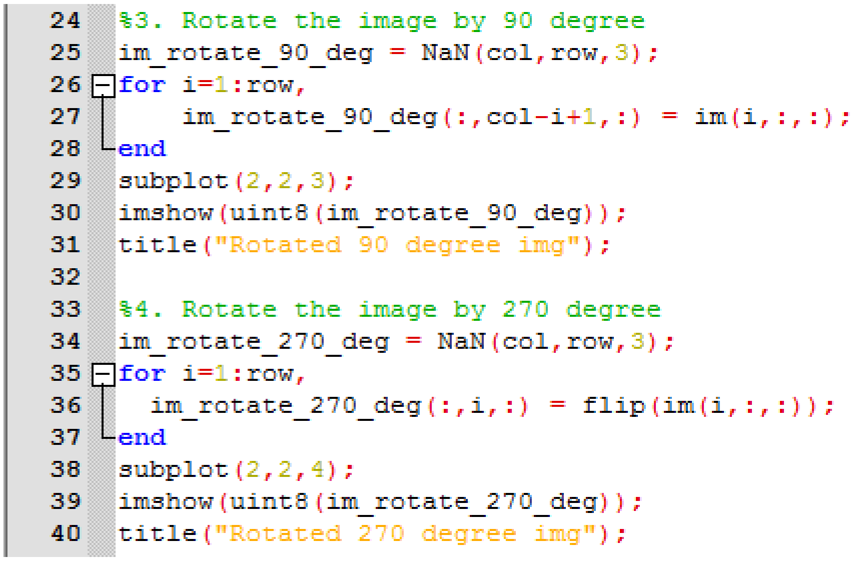
3) rotate the image by 90 degree

4)rotate the image by 270 degree

Code :

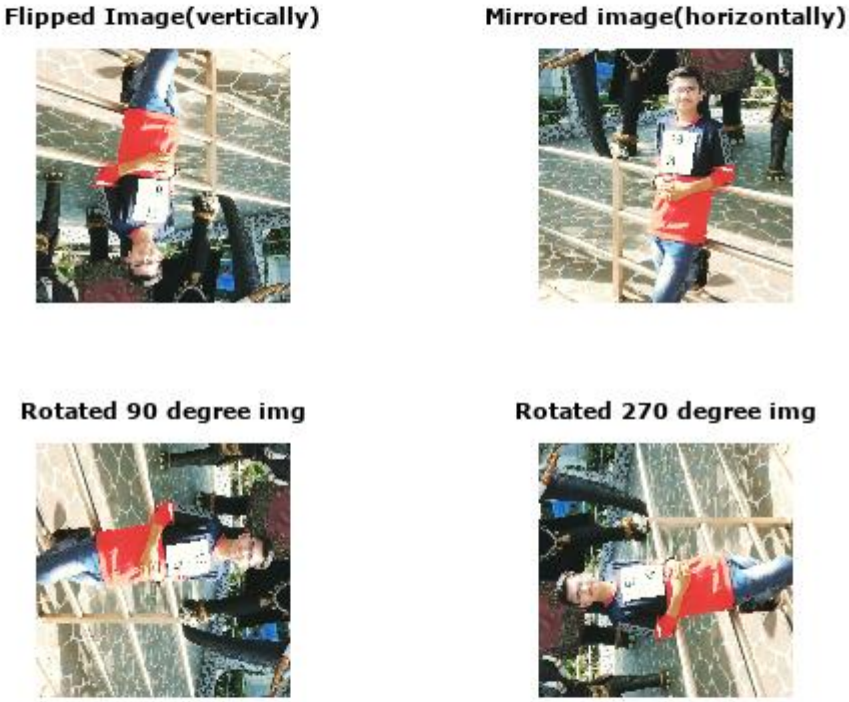


```
% Take your own photo
im = imread("my_image.jpg");

%1. Flip image vertically
row = size(im,1);
im_flipped_image = NaN(size(im));
for i=1:row,
    im_flipped_image(i,:,:) = im(row-i+1,:,:);
end
subplot(2,2,1);
imshow(uint8(im_flipped_image));
title("Flipped Image(vertically)")

%2. Create mirror image
col = size(im,2);
im_mirrored_image = NaN(size(im));
for j=1:col,
  im_mirrored_image(:,j,:) = im(:,col-j+1,:);
end
subplot(2,2,2);
imshow(uint8(im_mirrored_image));
title("Mirrored image(horizontally)");
```



```
%3. Rotate the image by 90 degree
im_rotate_90_deg = NaN(col,row,3);
for i=1:row,
    im_rotate_90_deg(:,col-i+1,:) = im(i,:,:);
end
subplot(2,2,3);
imshow(uint8(im_rotate_90_deg));
title("Rotated 90 degree img");

%4. Rotate the image by 270 degree
im_rotate_270_deg = NaN(col,row,3);
for i=1:row,
  im_rotate_270_deg(:,i,:) = flip(im(i,:,:));
end
subplot(2,2,4);
imshow(uint8(im_rotate_270_deg));
title("Rotated 270 degree img");
```

Output :



Flipped Image(vertically)

Mirrored image(horizontally)

Rotated 90 degree img

Rotated 270 degree img