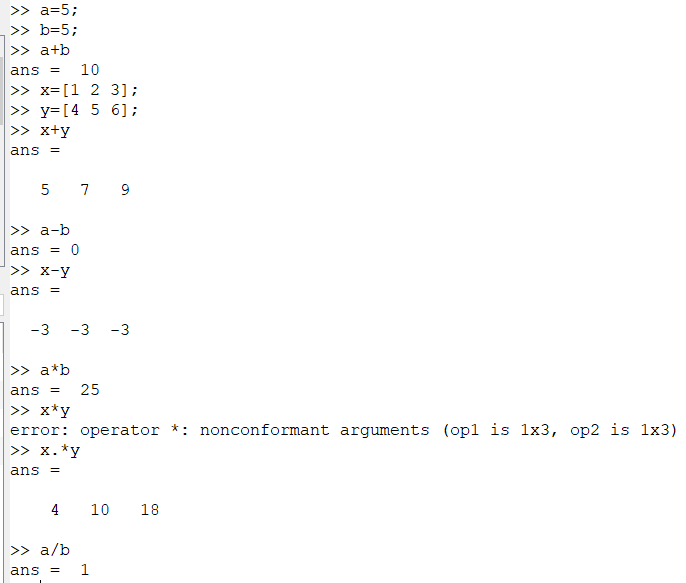
**IMAGE PROCESSING LAB-1**

**AIM: -** Introduction to Octave and its built-in functions

**Basic Operations: -**

* **Addition:** Used for the addition operation of variables, vectors, matrices, images
* **Subtraction:** Used for the subtracting variables, vectors, matrices, images
* **Multiplication:** Used for the multiplying variables, vectors, matrices, images. In matrix or vectors for multiplication”. \*” operand is used.
* **Division:** Used for the division operation of variables, vectors, matrices, images.



* For declaring vector(array) comma separated values of space separated values are used in [ ] brackets.

Example: >>A=[1,2,3]

* For declaring matrix each element of row is separated by space or comma and each row is separated by semicolon.

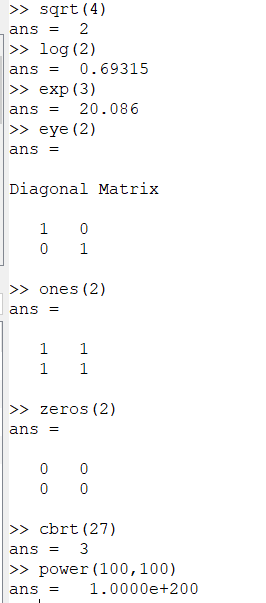
Example: >>x=[1 2 3;4 5 6]

>>y=[1,2,3;4,5,6]

* In Octave indices starts from 1.
* Help command is used to get documentation of any other command.
* clc command is used to clear command window.
* clear command is used to clear workspace. After performing clear all previous declared data are cleared.

**Built-in Functions: -**

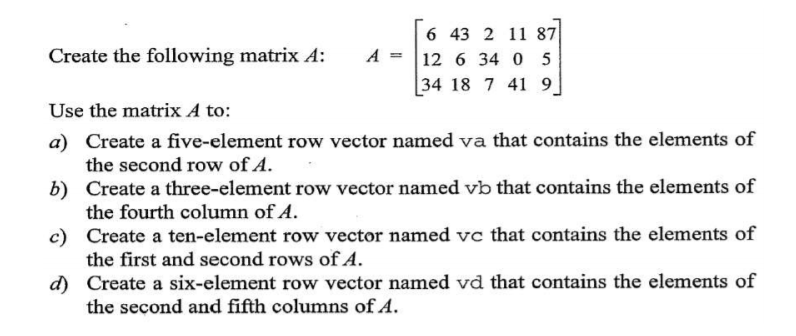
* **sqrt(n):** Used to get square root of n.
* **log(n):** Used calculate nature log of n.
* **exp(n):** Used calculate e^n.
* **eye(n):** It returns an identity matrix of dimension specified in n.
* **ones(n):** It returns a matrix filled with 1 at all position of dimension specified in n.
* **zeroes(n):** It returns a matrix filled with 0 at all position of dimension specified in n.
* **log10(n):** Used calculate log of n with base 10
* **cbrt(n):** Used to calculate cube root of n.
* **printf(n):** Used to print the given n string in command window.
* **power(a,b):** Used calculates a^b.



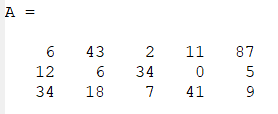
**Built-in Functions for images: -**

* **imread(n):** Usedto read an image from given filename n.
* **rgb2gray(n):** Usedconvert given rgb image n to gray scale image.
* **imwrite(n,m):** Usedto save image n with name m.
* **im2bw(n):** Usedconvert given rgb image n to black & white image.
* **imresize(n,m):** Usedchange the size of an image n to given in m.
* **figure:** Usedto open new window for displaying image.
* **title(n):** Usedto show the title in recently open image window.

**EXERSICE**

**Question 1:**

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



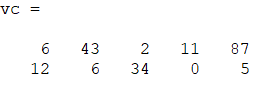
>>va = A(2,:)



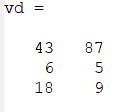
>>vb = A(:,4)



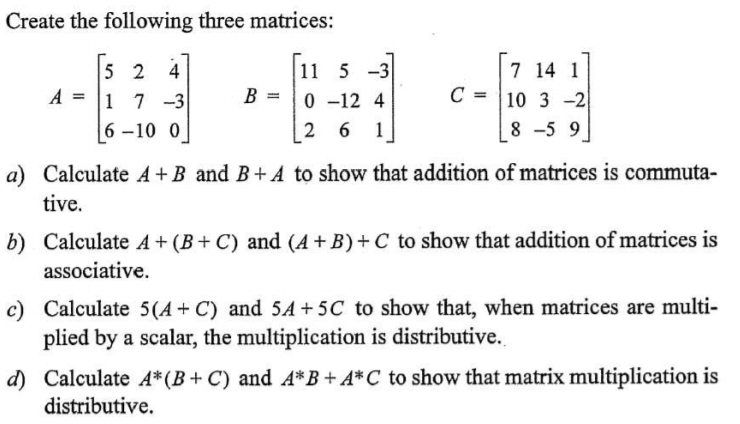
>>vc = A(1:2,:)



>>vd = A(:,2:3:5)



**Question 2:**



>> A=[5 2 4;1 7 -3;6 -10 0];

>> B=[11 5 -3; 0 -12 4;2 6 10];

>> C=[7 14 1;10 3 -2;8 -5 9];

1. **Code: -**

a1=A+B

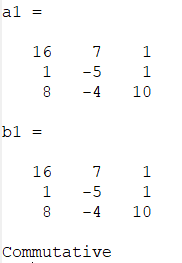
b1=B+A

if (a1==b1)

printf("Commutative\n");

endif

**Output: -**



1. **Code: -**

a2=A+(B+C)

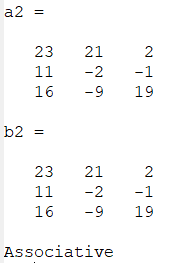
b2=(A+B)+C

if (a2==b2)

printf("Associative\n");

endif

**Output: -**



1. **Code: -**

a3=5\*(A+C)

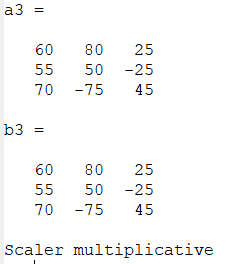
b3=(5\*A) + (5\*C)

if (a3=b3)

printf("Scaler multiplicative\n");

endif

**Output: -**



1. **Code: -**

a4=A\*(B+C)

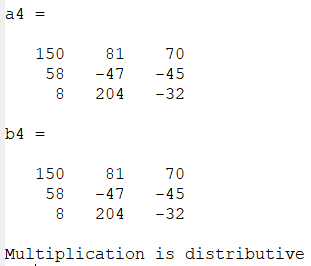
b4=(A\*B)+(A\*C)

if (a4==b4)

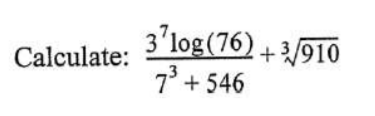
printf("Multiplication is distributive\n");

endif

**Output: -**



**Question 3:**



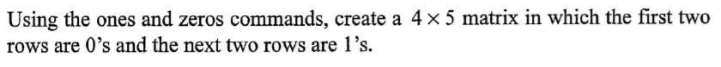
**Code: -**

>> res=((power(3,7)\*log(76))/(power(7,3)+546))+cbrt(910)

**Output: -**



**Question 4:**



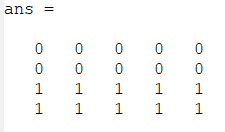
**Code: -**

>> p=zeros(2,5);

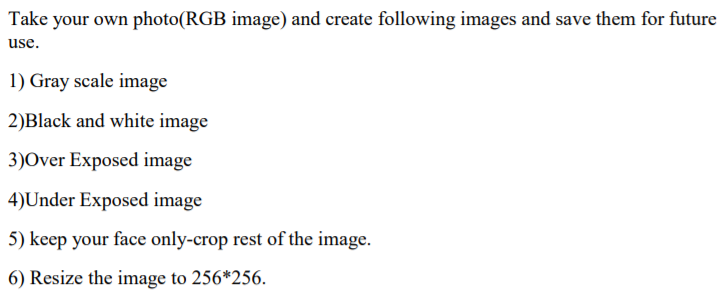
>> q=ones(2,5);

>> ans=[p;q]

**Output: -**



**Question 5:**



**Original Image: -**



1. **Gray Scale: -**

**Code: -**

pkg load image

img=imread('D:\IPLab\p1.jpg');

img\_gray=rgb2gray(img);

imwrite(img\_gray,"D:\IPLab\img\_gray.jpg");

**Ouput: -**



1. **Black and white: -**

**Code: -**

pkg load image

img=imread('D:\IPLab\p1.jpg');

img\_bw=im2bw(img);

imwrite(img\_bw,'D:\IPLab\img\_bw.jpg');

**Output:-**



1. **Over exposed: -**

**Code: -**

pkg load image

img=imread('D:\IPLab\p1.jpg');

img\_over=1.5\*img;

imwrite(img\_over,'D:\IPLab\imag\_over.jpg')

**Output: -**



1. **Under exposed: -**

**Code: -**

pkg load image

img=imread('D:\IPLab\p1.jpg');

img\_under=0.5\*img;

imwrite(img\_under,'D:\IPLab\img\_under.jpg')

**Output: -**



1. **Keep face only: -**

**Code: -**

pkg load image

img=imread('D:\IPLab\p1.jpg');

img\_face=img(200:750,500:970,:);

imwrite(img\_face,'D:\IPLab\img\_face.jpg');

**Output: -**



1. **Resize image: -**

**Code: -**

pkg load image

img=imread('D:\IPLab\p1.jpg');

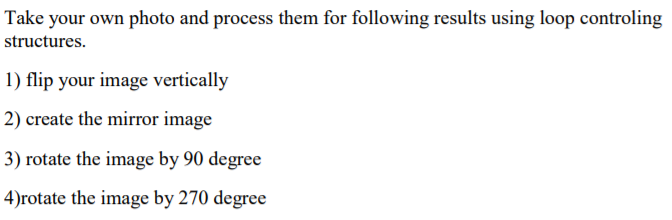
img\_resize=imresize(img,[256 256]);

imwrite(img\_resize,'D:\IPLab\img\_resize.jpg');

**Output: -**



**Question 6:**



1. **Flip image vertically: -**

**Code: -**

img = imread('D:\IPLab\p1.jpg');

[x, y, z] = size(img);

for plane = 1 : z

len = x;

for i = 1 : x

for j = 1 : y

if i < x/2

temp = img(i, j, plane);

img(i, j, plane) = img(len, j, plane);

img(len, j, plane) = temp;

end

end

len = len - 1;

end

end

imwrite(img,'D:\IPLab\fliped.jpeg')

**Output: -**



1. **Mirror image: -**

**Code:-**

pkg load image

img=imread('D:\IPLab\p1.jpg');

img\_mirror=uint8(zeros(size(img)));

for i=1:1391

for j=1:1800

img\_mirror(i,j,:)=img(i,1801-j,:);

endfor

endfor

imwrite(img\_mirror,'D:\IPLab\img\_mirror.jpg');

**Output: -**



1. **Rotate by 90 degree: -**

**Code: -**

pkg load image

img=imread('D:\IPLab\p1.jpg');

img\_90=uint8(zeros(1391,1800,3));

for i=1:1800

for j=1:1391

img\_90(i,j,:)=img(1801-j,i,:);

endfor

endfor

imwrite(img\_90,'D:\IPLab\img\_90.jpg');

**Output: -**



1. **Rotate by 270 degree: -**

**Code: -**

pkg load image

img=imread('D:\IPLab\p1.jpg');

img\_270=uint8(zeros(1391,1800,3));

for i=1:1800

for j=1:1391

img\_270(i,j,:)=img(j,1392-i,:);

endfor

endfor

imwrite(img\_270,'D:\IPLab\img\_270.jpg');

**Output: -**

