1.1- If x=[1 4; 8 3], find :

a) the inverse matrix of x .

b) the diagonal of x.

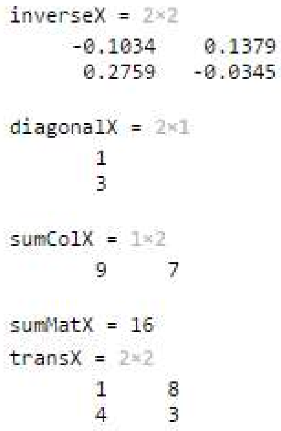
c) the sum of each column and the sum of whole matrix x.

d) the transpose of x.

Code:

```
clc;
clear all;
x = [1 4; 8 3];
% (a)inverse matrix of x
inverseX = inv(x)
% (b)diagonal of x
diagonalX = diag(x)
%% (c)sum of each column
sumColX = sum(x)
%% (c)sum of whole matrix
sumMatX = sum(sum(x))
%% (d)transpose of x
transX = x'
```

Output:



```
inverseX = 2×2
     -0.1034    0.1379
      0.2759   -0.0345

diagonalX = 2×1
       1
       3

sumColX = 1×2
       9     7

sumMatX = 16
transX = 2×2
       1     8
       4     3
```

1.2- If x= [2 8 5; 9 7 1], b=[2 4 5] find:

a) Find the maximum and minimum of x.

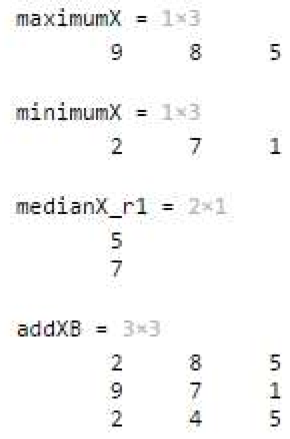
b) Find median value over each row of x.

c) Add the vector b as a third row to x.

Code:

```
clc;
clear all;
x= [2 8 5; 9 7 1];
b=[2 4 5];
%% (a)maximum and minimum of x
maximumX = max(x)
minimumX = min(x)
%% (b)median value over each row of x
medianX_r1 = median(x,2)
%% (c)adding vector b as third row of x
addXB = [x;b]
```

Output:



```
maximumX = 1×3
     9     8     5

minimumX = 1×3
     2     7     1

medianX_r1 = 2×1
     5
     7

addXB = 3×3
     2     8     5
     9     7     1
     2     4     5
```

1.3- If x=[2 6 12; 15 6 3; 10 11 1], then

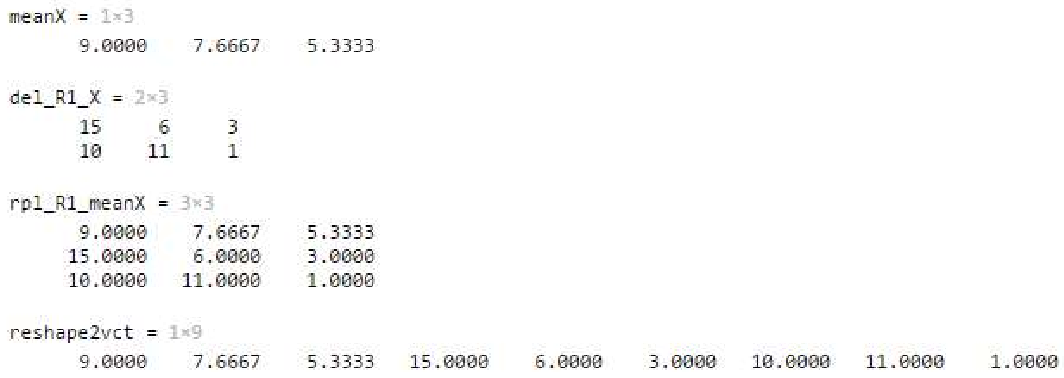
a) replace the first row elements of matrix x with its average value.

b) reshape this matrix into row vector.

Code:

```
clc;

clear all;

x = [ 2 6 12; 15 6 3; 10 11 1];

%% (a)replacing the 1st row elements of mat X with its avg val

meanX = mean(x)

x(1:1,:) = [];

del_R1_X = x

rpl_R1_meanX = [meanX; del_R1_X]

%% (b)reshaping this matrix into row vector

y = rpl_R1_meanX;

reshape2vct = [y(1:1,1:3) y(2:2,1:3) y(3:3,1:3)]
```

Output:



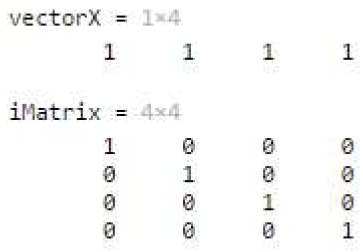
```
meanX = 1×3
      9.0000    7.6667    5.3333

del_R1_X = 2×3
      15     6     3
      10    11     1

rpl_R1_meanX = 3×3
      9.0000    7.6667    5.3333
     15.0000    6.0000    3.0000
     10.0000   11.0000    1.0000

reshape2vct = 1×9
      9.0000    7.6667    5.3333   15.0000    6.0000    3.0000   10.0000   11.0000    1.0000
```

1.4- Generate a 4 x 4 Identity matrix.

Code:

```
clc;

clear all;

vectorX = ones(1,4)

%% generating 4x4 identity matrix

iMatrix = diag(vectorX)
```

Output:



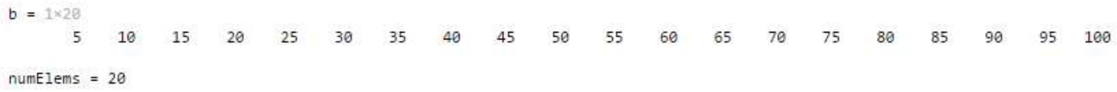
```
vectorX = 1×4
       1     1     1     1

iMatrix = 4×4
       1     0     0     0
       0     1     0     0
       0     0     1     0
       0     0     0     1
```

1.5- Generate the following row vector, b=[5, 10, 15, 20 … 95, 100], then find the number of elements in this vector.

Code:

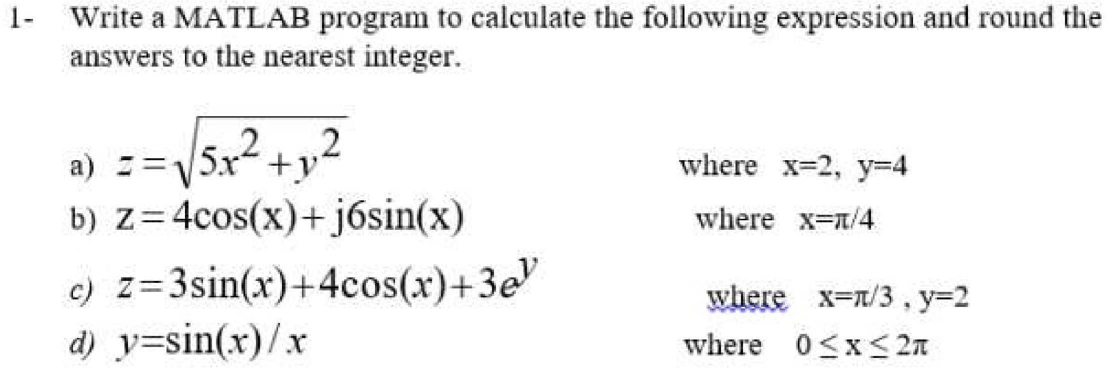
```
clc;

clear all;

%% generating the given vector

b = [5:5:100]

%% finding the number of elements of the vector

numElems = length(b)
```

Output:



```
b = 1×20
       5    10    15    20    25    30    35    40    45    50    55    60    65    70    75    80    85    90    95   100

numElems = 20
```

2.1



1- Write a MATLAB program to calculate the following expression and round the answers to the nearest integer.

a) $z=\sqrt{5x^2+y^2}$ where x=2, y=4

b) $z=4\cos(x)+j6\sin(x)$ where x=π/4

c) $z=3\sin(x)+4\cos(x)+3e^{y}$ where x=π/3 , y=2

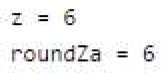
d) $y=\sin(x)/x$ where $0\leq x\leq 2\pi$

Code:

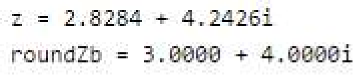
clc;

```
clear all;
%% (a)
x = 2;
y = 4;
z = (sqrt(5*(x^2)+(y^2)))
roundZa = round(z)
%% (b)
x = pi/4;
z = (4*(cos(x)) + j*6*(sin(x)))
roundZb = round(z)
%% (c)
x = pi/3;
y = 2;
z = (3*(sin(x)) + 4*(cos(x)) + 3*(exp(y)))
roundZc = round(z)
%% (d)
x = 0:(2*pi);
y = ((sin(x))/x)
roundYd = round(y)
```

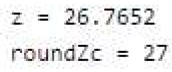
Output:



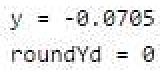
```
z = 6
roundZa = 6
```



```
z = 2.8284 + 4.2426i
roundZb = 3.0000 + 4.0000i
```



```
z = 26.7652
roundZc = 27
```



```
y = -0.0705
roundYd = 0
```

2.2- Solve the following system

x + y - 2z = 3

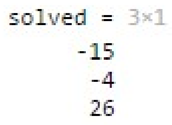
2x + y = 7

x + y - z = 4

Code:

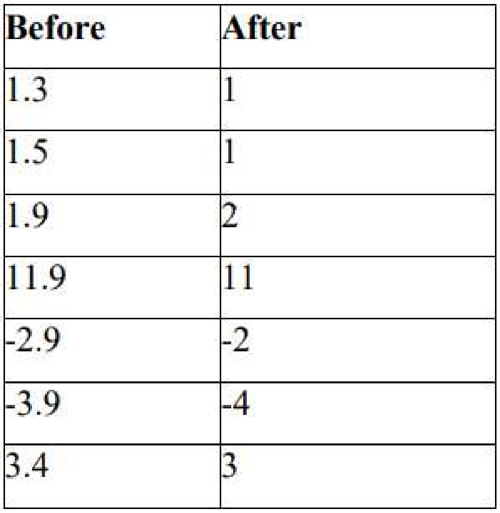
```
clc;

clear all;

%% solving the given system

a = [1 2 1; 1 1 1; -2 0 -1];

b = [3; 7; 4];

solved = inv(a)*b
```

Output:



```
solved = 3×1
     -15
      -4
      26
```

2.3- Use [round, fix, ceil, floor] commands to round the following numbers towards integer numbers:



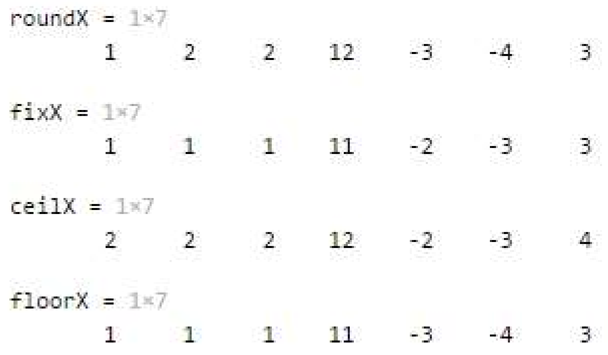
Before	After
1.3	1
1.5	1
1.9	2
11.9	11
-2.9	-2
-3.9	-4
3.4	3

Code:

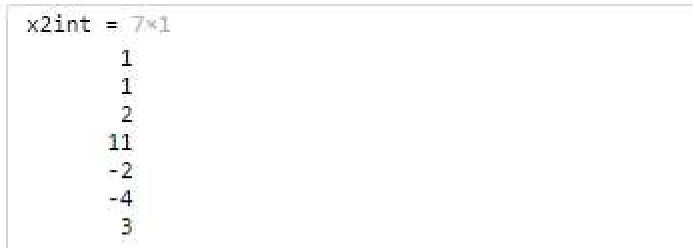
```
clc;
clear all;
x = [1.3 1.5 1.9 11.9 -2.9 -3.9 3.4];
%% round
roundX = round(x)
%% fix
fixX = fix(x)
%% ceil
ceilX = ceil(x)
%% floor
floorX = floor(x)
%% round, fix, ceil, floor
x2int = [round(1.3);
 fix(1.5);
round(1.9);
 fix(11.9);
 ceil(-2.9);
 floor(-3.9);
```

floor(3.4)]

Output:



```
roundX = 1×7
       1     2     2    12    -3    -4     3

fixX = 1×7
       1     1     1    11    -2    -3     3

ceilX = 1×7
       2     2     2    12    -2    -3     4

floorX = 1×7
       1     1     1    11    -3    -4     3
```



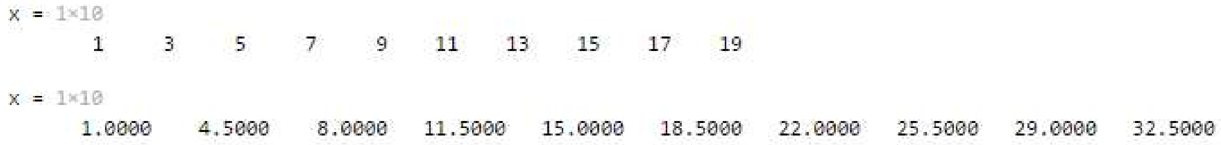
```
x2int = 7×1
       1
       1
       2
      11
      -2
      -4
       3
```

2.4- Generate ten values from the uniform distribution on the interval [2, 3.5].

Code:

```
clc;

clear all;

interval_1 = 2;

interval_2 = 3.5;

%% interval 2

x = 1 : interval_1 : interval_1*10

%% interval 3.5

x = 1 : interval_2 : interval_2*10
```

Output:



```
x = 1×10
      1     3     5     7     9    11    13    15    17    19

x = 1×10
     1.0000    4.5000    8.0000   11.5000   15.0000   18.5000   22.0000   25.5000   29.0000   32.5000
```

3.1- write a program to read three bits x, y, z, then compute:

a) v = (x and y) or z

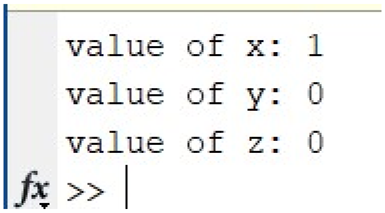
b) w = not (x or y) and z

c) u = (x and not (y)) or (not (x) and y)

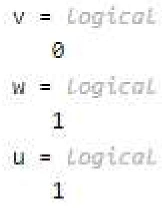
Code:

```
clc;

clear all;

%% reading the bits of x, y, z

x = input("value of x: ");

y = input("value of y: ");

z = input("value of z: ");

%% computing (a)

v = or(and(x, y), z)

%% computing (b)

w = ~(and(or(x, y), z))

%% computing (c)

u = or(and(x, ~(y)), and(~(x), y))
```

Output:



```
value of x: 1
value of y: 0
value of z: 0
fx >>
```



```
v = logical
   0
w = logical
   1
u = logical
   1
```

3.3- if q=[1 5 6 8 3 2 4 5 9 10 1],x=[3 5 7 8 3 1 2 4 11 5 9],

then:

a) find elements of (q) that are greater than 4.

b) find elements of (q) that are equal to those in (x).

c) find elements of (x) that are less than or equal to 7.

Code:

clc;

clear all;

q = [1 5 6 8 3 2 4 5 9 10 1];

x=[3 5 7 8 3 1 2 4 11 5 9];

%% (a)finding elements of q that are greater than 4

qHigh = q>4

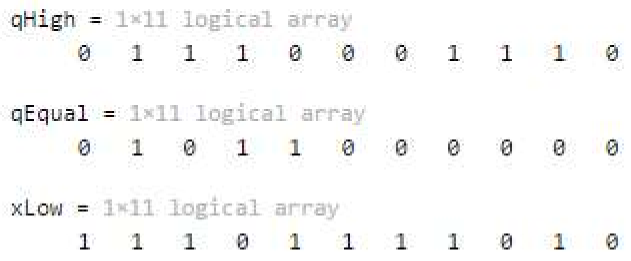
%% (b)finding elements of q thar are equal to those in x

qEqual = q==x

%% (c)finding elements of x that are less than or equal to 7

xLow = x<=7

Output:



```
qHigh = 1×11 logical array
     0   1   1   1   0   0   0   1   1   1   0

qEqual = 1×11 logical array
     0   1   0   1   1   0   0   0   0   0   0

xLow = 1×11 logical array
     1   1   1   0   1   1   1   1   0   1   0
```

3.4- If x=[10 3 ; 9 15], y=[10 0; 9 3], z=[-1 0; -3 2], what is the output of the following statements:

a) v = x > y

b) w = z >= y

c) u = ~z & y

d) t = x & y < z

Code:

clc;

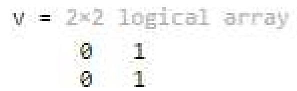
clear all;

x = [10 3; 9 15];

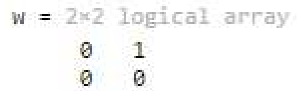
y = [10 0; 9 3];

```
z = [-1 0; -3 2];

%% (a)

v = x > y

%% (b)

w = z >= y

%% (c)

u = and(~(z),y)

%% (d)

t = and(x,y) < z
```

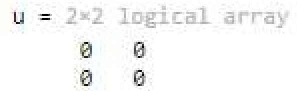
Output:



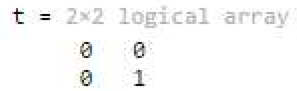
```
v = 2×2 logical array
      0   1
      0   1
```



```
w = 2×2 logical array
      0   1
      0   0
```

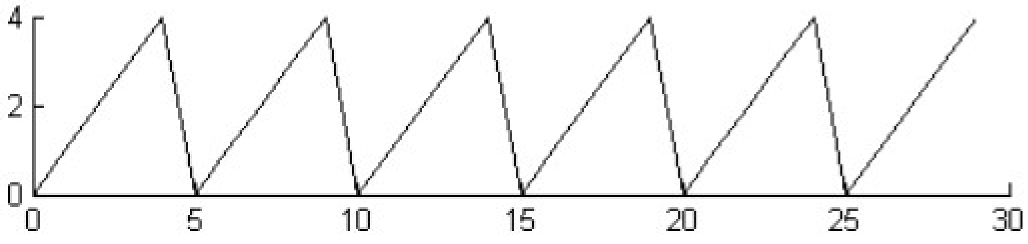


```
u = 2×2 logical array
      0   0
      0   0
```



```
t = 2×2 logical array
      0   0
      0   1
```

4.1- Plot sawtooth waveform as shown below

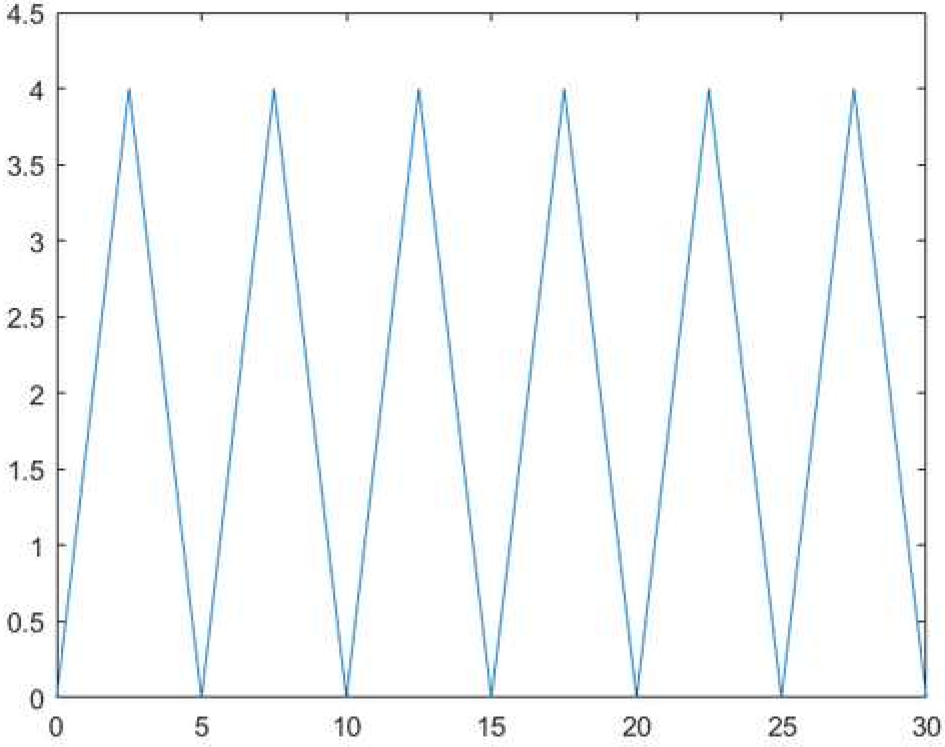


Code

```
clc;

clear all;

t = 0:2.5:30;
```

```
x = 4*sawtooth(pi*5*2*t)+4;
plot(t,x)
```

Output:

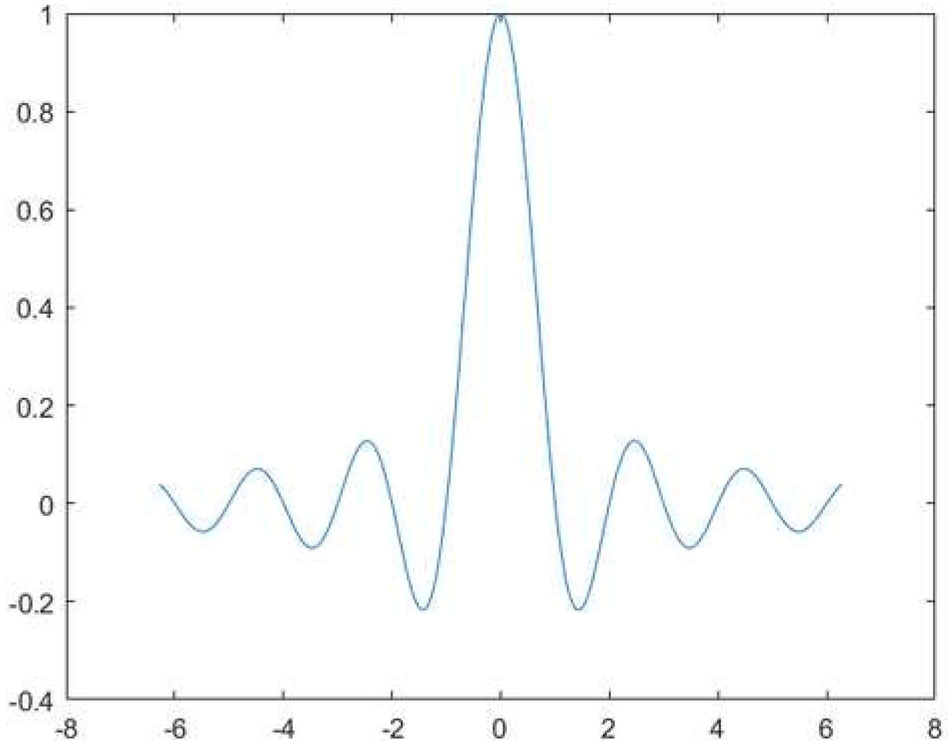


4.2- Plot Sinc function, where Sinc (x) = sin(x) / x , and -2π ≤ x ≤ 2π

Code:

```
clc;
clear all;
x = -2*pi:pi/50:2*pi;
plot(x, sinc(x))
```

Output:



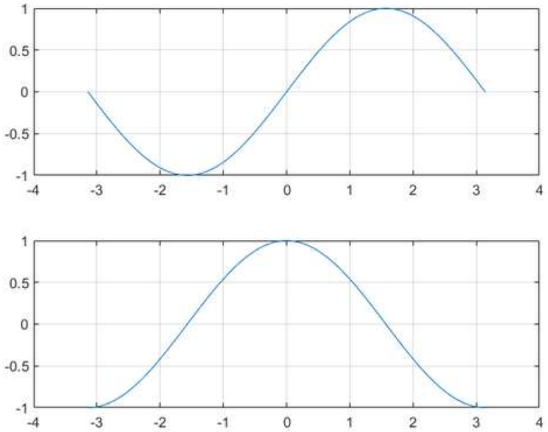
4.3- Plot sin(x) and cos(x) on the same figure, then on the same axis using different colors.

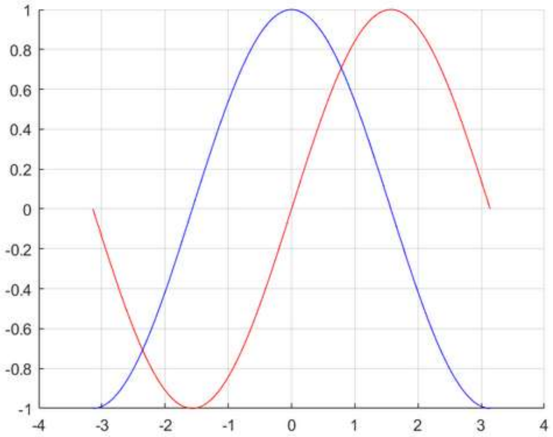
Code:

```
clc;
clear all;
x = -pi:pi/100:pi;
%% plotting on the same graph
figure(1)
subplot(2,1,1)
plot(x, sin(x))
grid
subplot(2,1,2)
plot(x, cos(x))
grid
%% plotting on same axis
```

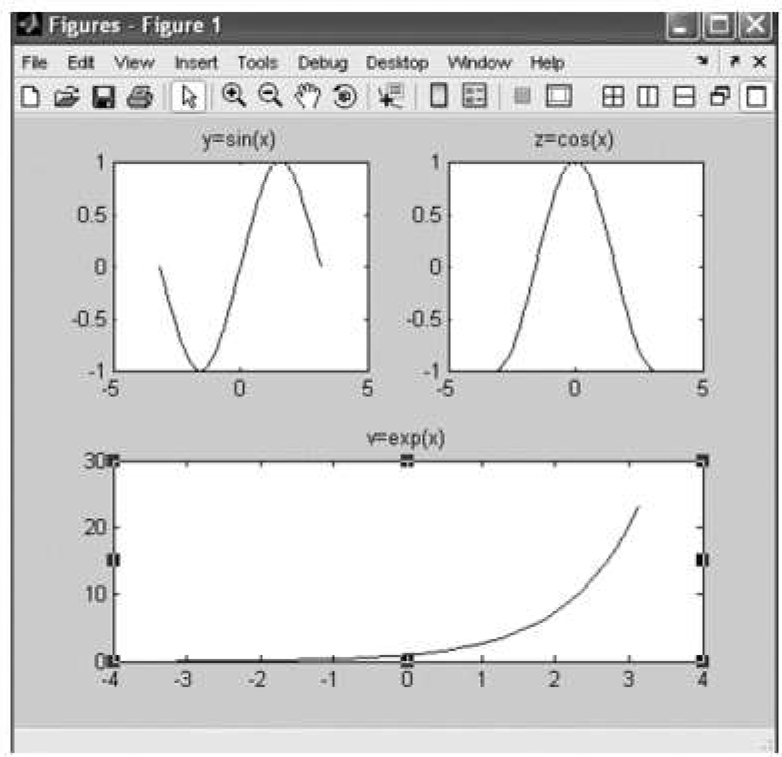
```
figure(2)
hold on
plot(x, sin(x), 'r')
plot(x, cos(x), 'b')
hold off
grid
```

Output:





4.4- if y=sin(x), z=cos(x), v=exp(x), where -π ≤ x ≤ π. Could you plot y, z, v as shown below!

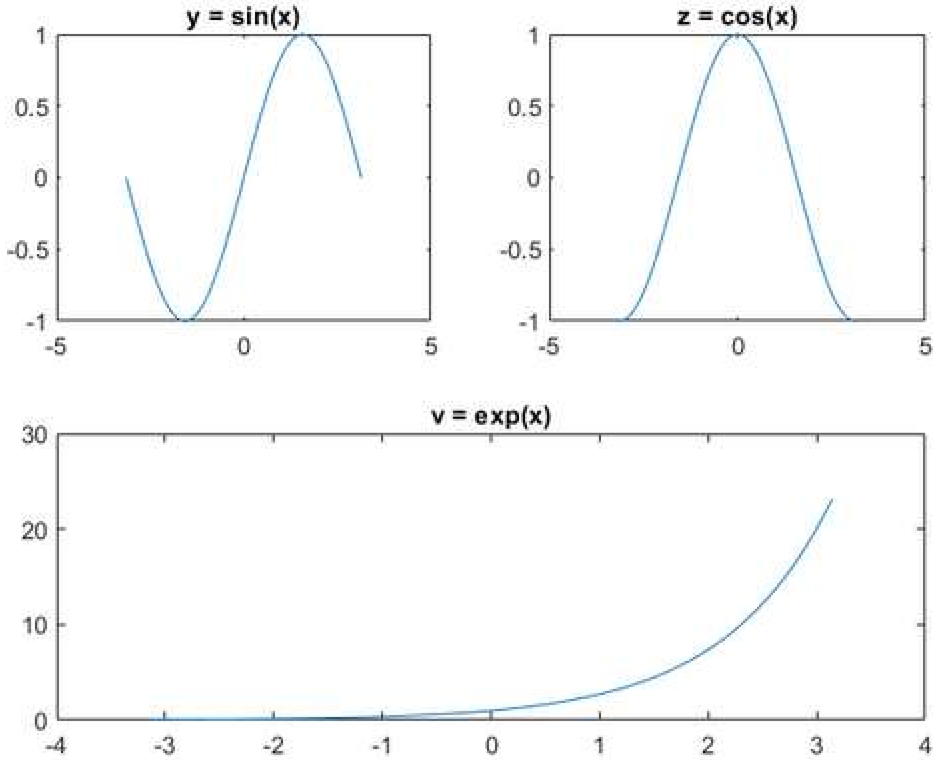


Code:

```
clc;
clear all;
x = -pi:(pi/100):pi;
y = sin(x);
subplot(2,2,1)
plot(x, y)
title('y = sin(x)')
axis([-5 5 -1 1])
z = cos(x);
subplot(2,2,2)
plot(x, z)
title('z = cos(x)')
axis([-5 5 -1 1])
v = exp(x);
```

```
subplot(2,1,2)

plot(x, v)

title('v = exp(x)')

axis([-4 4 0 30])
```

Output:



5.1- Represent the following complex numbers in polar coordinate

Z= 2 + 5j

Y= -3 - 3j

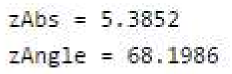
D= -2 + 6j

Code:

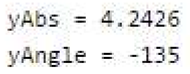
```
clc;

clear all;

%% Z

Z = 2+5*j;
```

```
zAbs = abs(Z)

zAngle = angle(Z)*180/pi

%% Y

Y = -3-3*j;

yAbs = abs(Y)

yAngle = angle(Y)*180/pi

%% D

D = -2+6*j;

dAbs = abs(D)

dAngle = angle(D)*180/pi
```

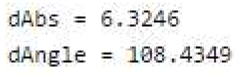
Output:



```
zAbs = 5.3852
zAngle = 68.1986
```



```
yAbs = 4.2426
yAngle = -135
```



```
dAbs = 6.3246
dAngle = 108.4349
```

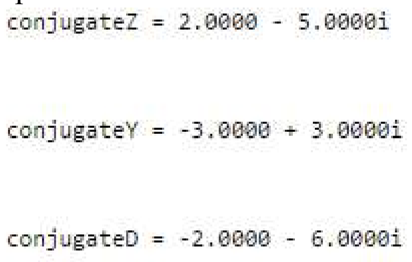
5.2- Find the conjugate of the numbers above.

Code:

```
clc;

clear all;

%% Z

Z = 2+5*j;

conjugateZ = conj(Z)

%% Y

Y = -3-3*j;

conjugateY = conj(Y)

%% D
```

```
D = -2+6*j;

conjugateD = conj(D)
```

Output:



```
conjugateZ = 2.0000 - 5.0000i


conjugateY = -3.0000 + 3.0000i


conjugateD = -2.0000 - 6.0000i
```

5.3- Represent the following numbers in rectangular coordinate

W= 5∟30o

A= 2.5∟-20o

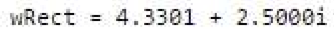
Q= 3e1.5∟-73o

Code:

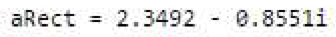
```
clc;

clear all;

%% W in rectangle

wAbs = 5;

wAngle = 30*(pi/180);

[x, y] = pol2cart(wAngle, wAbs);

wRect = complex(x, y)

%% A in rectangle

aAbs = 2.5;

aAngle = -20*(pi/180);

[x, y] = pol2cart(aAngle, aAbs);

aRect = complex(x, y)

%% Q in rectangle

qAbs = 3*exp(1.5);

qAngle = -73*(pi/180);

[x, y] = pol2cart(qAngle, qAbs);
```

qRect = complex(x, y)

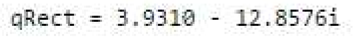
Output:



```
wRect = 4.3301 + 2.5000i
```



```
aRect = 2.3492 - 0.8551i
```



```
qRect = 3.9310 - 12.8576i
```

5.4- Write a program to compute the most frequent numbers in vectors (x), and (y) if

x= a*b y=a* c

a = [1 3]

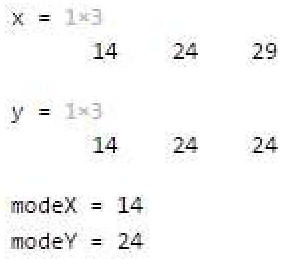
b = [2 3 5 ; 4 7 8]

c = [2 3 3 ; 4 7 7]

Code:

```
clc;
clear all;
a = [1 3];
b = [2 3 5; 4 7 8];
c = [2 3 3; 4 7 7];
x = a*b
y = a*c
modeX = mode(x)
modeY = mode(y)
```

Output:



```
x = 1×3
      14    24    29

y = 1×3
      14    24    24

modeX = 14
modeY = 24
```

6.1- If x=[1 5 9; 2 7 4], then

a) display the last two elements by using disp command.

b) display the sum of each row as show below

The sum of 1st row =

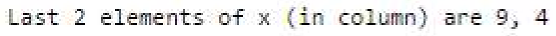
The sum of 2nd row =

Code:

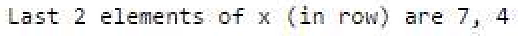
```
clc;
clear all;
x = [1 5 9;
 2 7 4];
%% (a)displaying last 2 elements
% in last column
colEnd1 = num2str(x(end-1));
colEnd2 = num2str(x(end));
disp(['Last 2 elements of x (in column) are ' colEnd1 ', ' colEnd2])
% in last row
rowEnd1 = num2str(x(4));
rowEnd2 = num2str(x(6));
disp(['Last 2 elements of x (in row) are ' rowEnd1 ', ' rowEnd2])
%% (b)displaying sum of each row
sumRows = sum(x,2);
sumRow1 = num2str(sumRows(1));
sumRow2 = num2str(sumRows(2));
```

```
disp(['The sum of 1st row = ' sumRow1])

disp(['The sum of 2nd row = ' sumRow2])
```

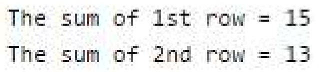
Output:



```
Last 2 elements of x (in column) are 9, 4
```



```
Last 2 elements of x (in row) are 7, 4
```



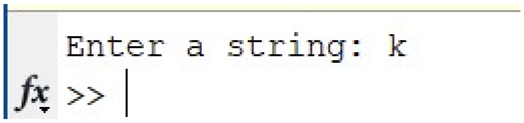
```
The sum of 1st row = 15
The sum of 2nd row = 13
```

6.2- Write a program to read a string, then replace each character in the string with its following character in ASCII code*.

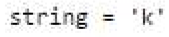
Code:

```
clc;

clear all;

%% reading string

string = input('Enter a string: ', 's')

%% converting to ascii

strAscii = num2str(double(string));

disp(['Ascii value of each character: ' strAscii])
```

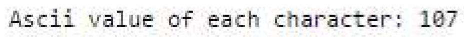
Output:



```
Enter a string: k
fx >> 
```

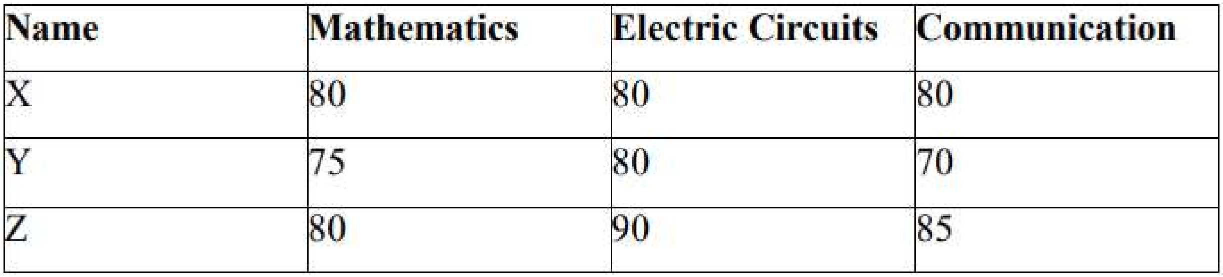


```
string = 'k'
```

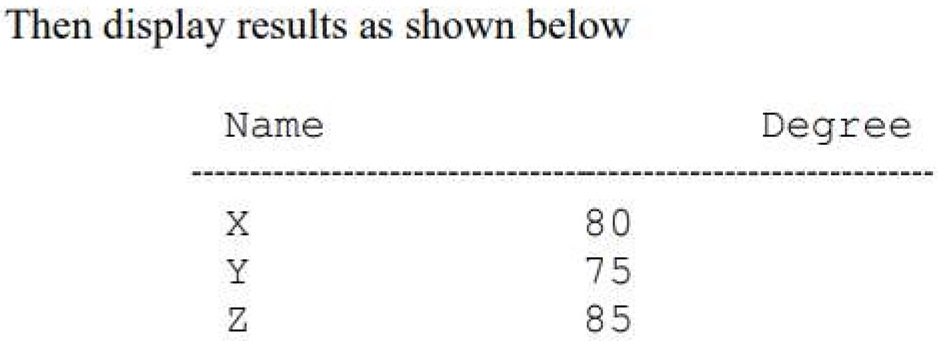


```
Ascii value of each character: 107
```

6.3- The Table shown below lists the degrees of three students, Write a program in M-file to read these degrees and calculate the average degree for each student.



Name	Mathematics	Electric Circuits	Communication
X	80	80	80
Y	75	80	70
Z	80	90	85



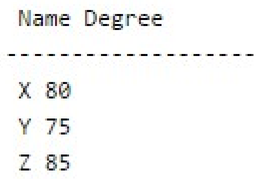
Then display results as shown below

```
Name                    Degree
-----------------------------------------------------------
X                 80
Y                 75
Z                 85
```

Code:

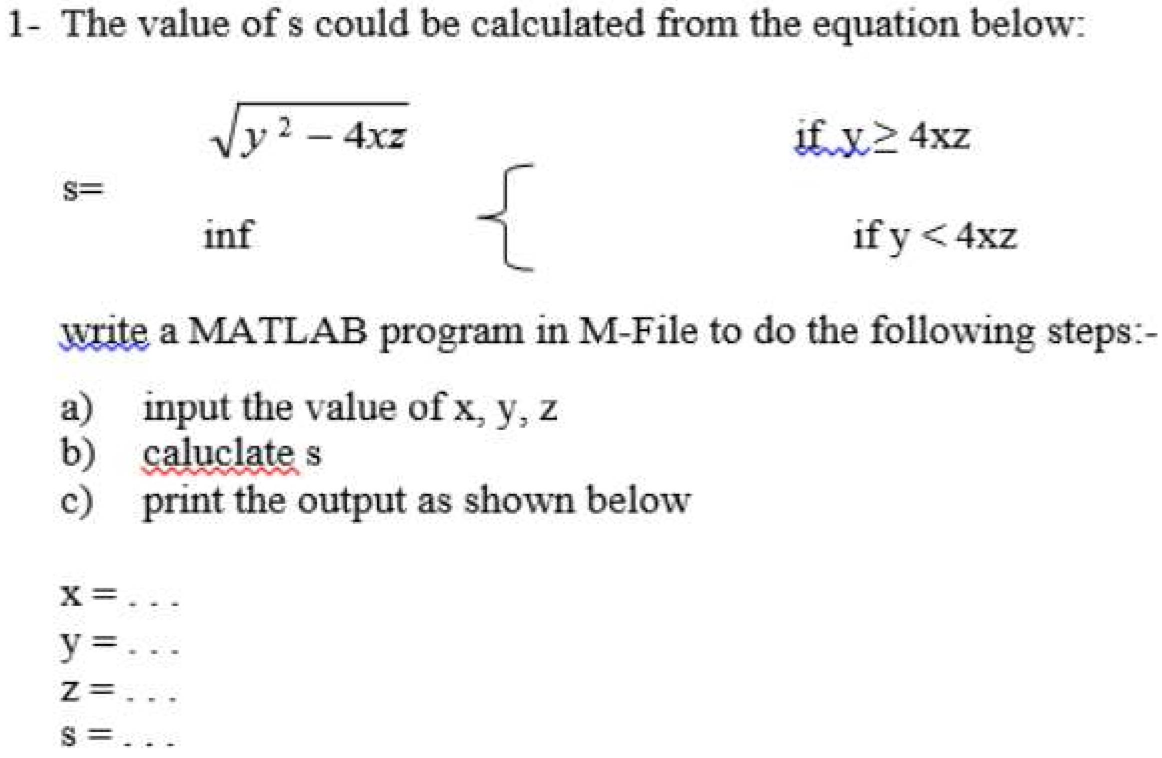
```
clc;

clear all;

X = [80 80 80];

sumX = sum(X,2);

avgX = num2str(sumX/length(X));

Y = [75 80 70];

sumY = sum(Y,2);

avgY = num2str(sumY/length(Y));

Z = [80 90 85];

sumZ = sum(Z,2);

avgZ = num2str(sumZ/length(Z));

disp([' Name' ' ' 'Degree'])

disp(['-------------------'])

disp([' X' ' ' avgX])

disp([' Y' ' ' avgY])
disp([' Z' ' ' avgZ])
```

Output:



```
 Name Degree
-------------------
 X 80
 Y 75
 Z 85
```

7.1



1- The value of s could be calculated from the equation below:

$$s= \begin{array}{lcr} \sqrt{y^2-4xz} & & \text{if } y \geq 4xz \\ \text{inf} & \Big\{ & \text{if } y < 4xz \end{array}$$

write a MATLAB program in M-File to do the following steps:-

a) input the value of x, y, z

b) caluclate s

c) print the output as shown below

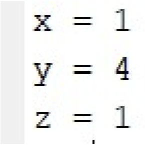
x = . . .
y = . . .
z = . . .
s = . . .

Code:

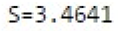
```
clc;
clear all;
%% (a)input the value of x, y, z
x = input('x = ');
y = input('y = ');
z = input('z = ');
%% (b)calculate s
if y>=4*x*z
```

```
 s= sqrt(y^2-4*x*z);
else
 s= inf;
end
disp(['S=' num2str(s)])
```

Output:



```
x = 1
y = 4
z = 1
```



```
S=3.4641
```