Data Mining

HW 1

Atieh Kashani

1.2.

a.
```
>> a=[1 2 3; 4 5 6];
>> b=zeros(size(a));
```

Creates a matrix 2x3 of zeros (2 rows and 3 columns that is the size of matrix ‘a’)

```
b =
    0    0    0
    0    0    0
```

b.
```
>> x=round([1.5 2; 2.2 3.1])
>> a=find(x(:)>2);
```

Rounding the matrix to the nearest integer, x= [2 2;2 3]
Returns the index of the element of ‘x’ which is greater than 2

```
a = 4
```

c.
```
>> a = [1 2; 3 4; 5 6];
>> b=a(:);
>> c=reshape(b, 6,1)
```

d.

b=> a 6x1 column vector (6 rows and 1 column) by stacking up of columns of ‘a’
c=>Makes a 6x1 matrix (vector) out of vector ‘b’ elements
b and c return similar results

```
c =
    1
    3
    5
    2
    4
    6
```

e.
```
>>a=rand(1, 100);
>>b=randperm(100);
>>c=a(b(1:5));
```

a=> Randomly fill ‘a’ that is a row vector with 100 elements that are normalized values in the interval (0,1)
b => a row vector containing a random permutation of the integers from 1 to 100
b(1:5) => Returns the elements of 1 through 5 of ‘b’ that are between 1 and 100
Output: c=a(b(1:5)) => Returns the elements of row vector ‘a’ which their indices specified from this command ‘b(1:5)’

Example output:
c = 0.4795 0.8611 0.1476 0.0900 0.5861

f.
```
>>a=[100:200];
>>b=find(a>120);
>>c=a(b);
```

Create a row vector ‘a’ with elements between 100 and 200 (both are included)

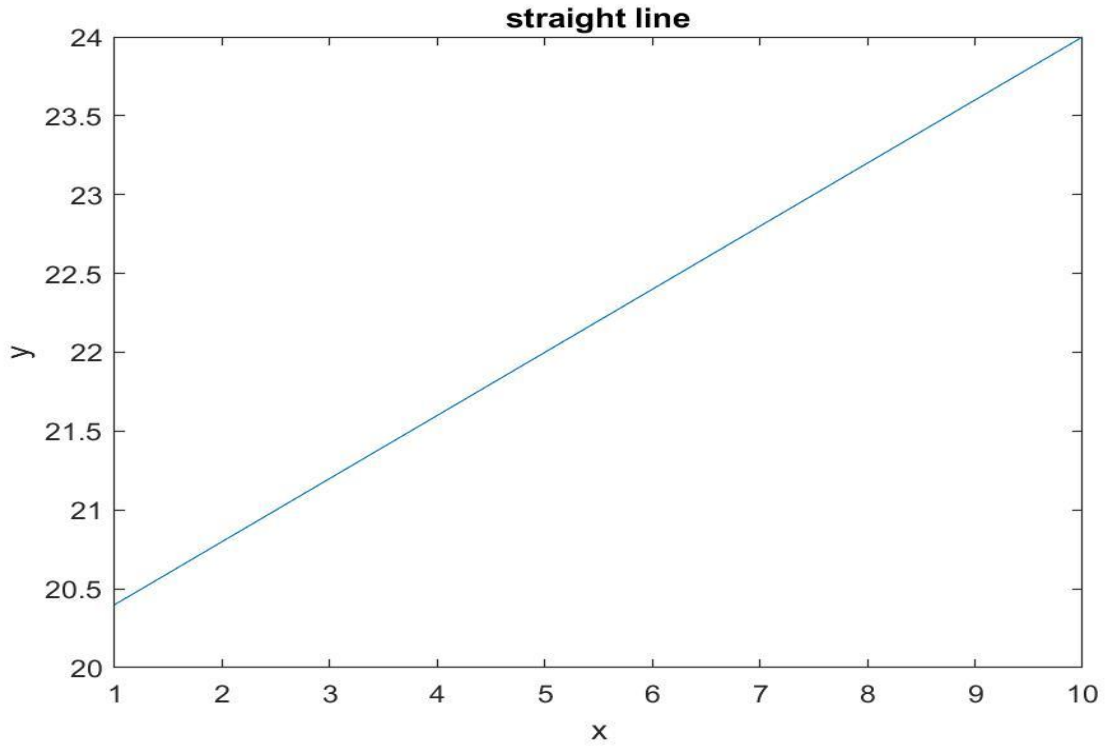
Find indices of the elements of ‘a’ that are greater than 120

Output: c => elements of ‘a’ that are greater than 120 (the indices specified in the previous command) which are 121 to 200

1.3. Select a proper range for the variable x, and visualize the curves for the following functions:

- straight line $y = \theta_0 + \theta_1 x$ where $\theta_0 = 20,\ \theta_1 = 0.4$

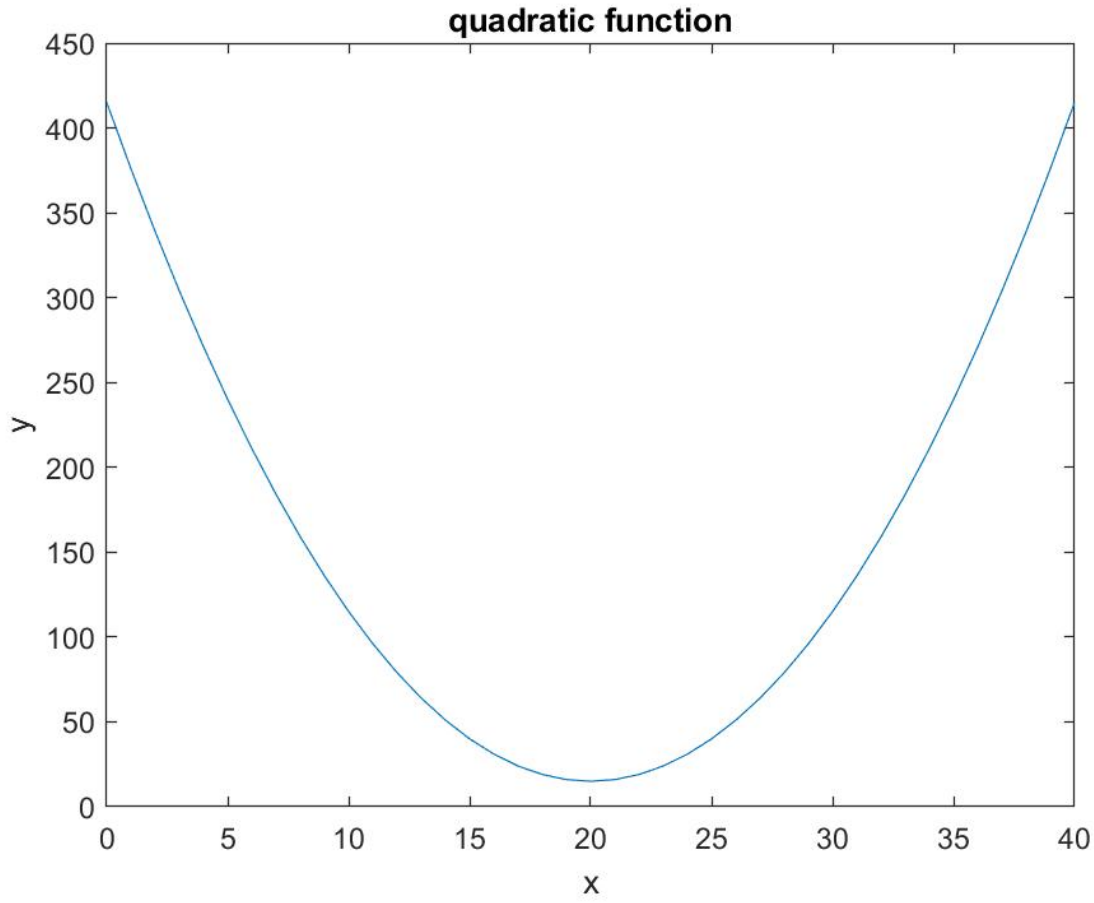
Code:
```
x=1:10;
t0=20;
t1=0.4;
y=t0+t1*x;
plot(x,y)
title('straight line');
xlabel('x');
ylabel('y');
```



- quadratic function: $y = (x - \theta_1)^2 + \theta_0, where\ \theta_1 = 20, \theta_0 = 15$

Code:

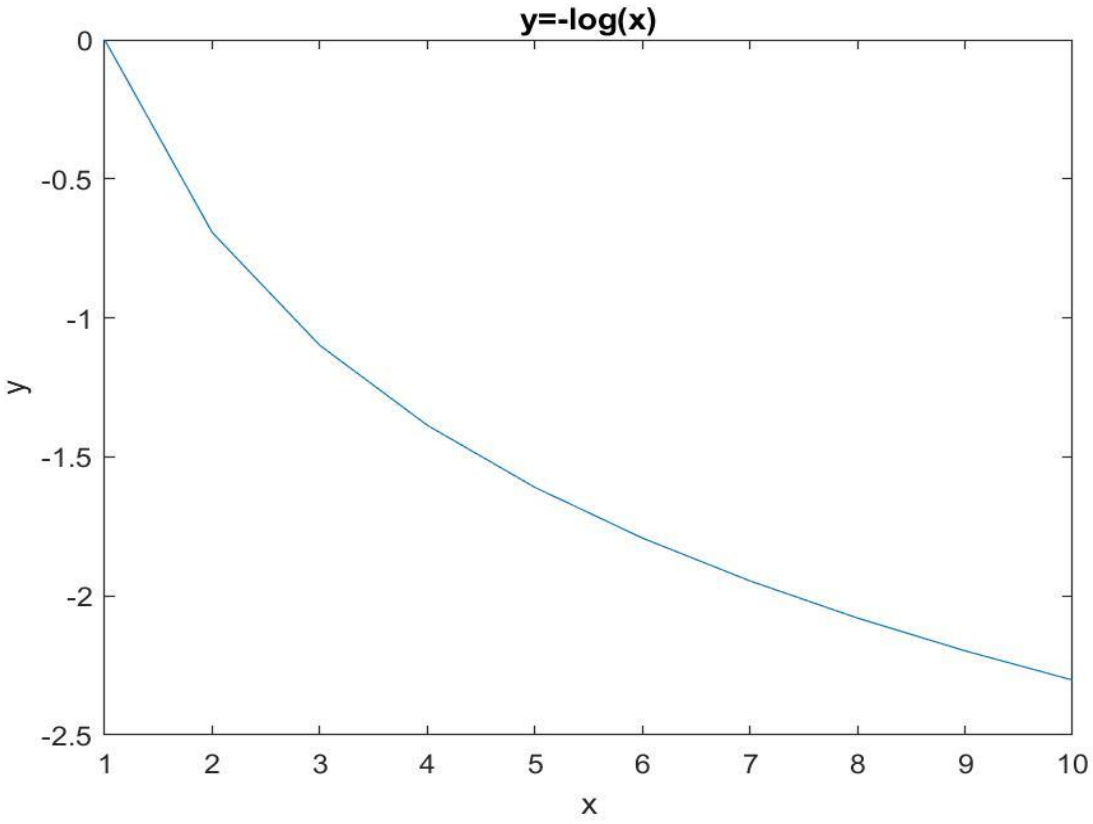
```
x=0:40;
t0= 15;
t1= 20;
y = (x-t1).^2 + t0;
plot(x,y)
title('quadratic function');
xlabel('x');
ylabel('y');
```



- log function, $y = -\log(x)$ and $y = -\log(1 - x)$

code:

```
x=1:10;
y=-log(x);
plot(x,y)
title('y=-log(x)');
xlabel('x');
ylabel('y');
```

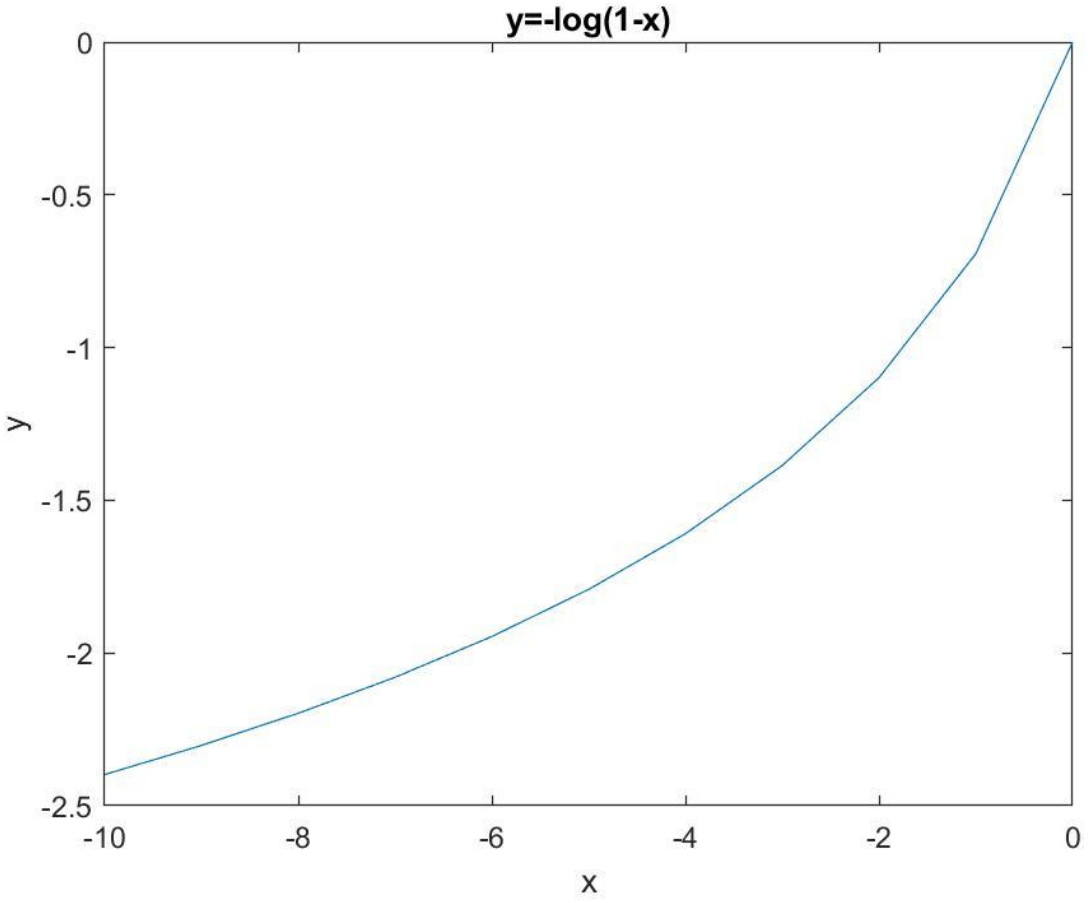


- $y = -\log(1-x)$

Code:

```
x=-10:0;
y=-log(1-x);
```

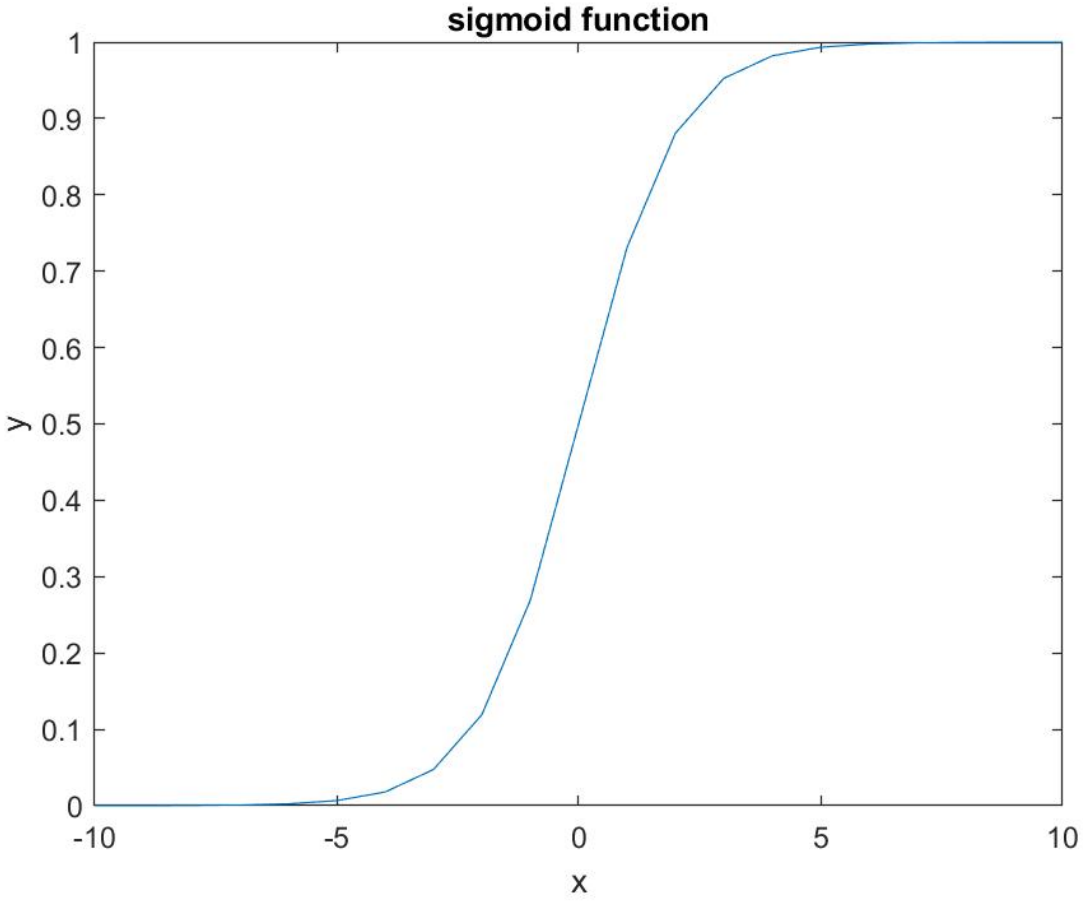
```
plot(x,y)
title('y=-log(1-x)');
xlabel('x');
ylabel('y');
```



- sigmoid function, $y = 1/(1 + e^{-x})$

Code:

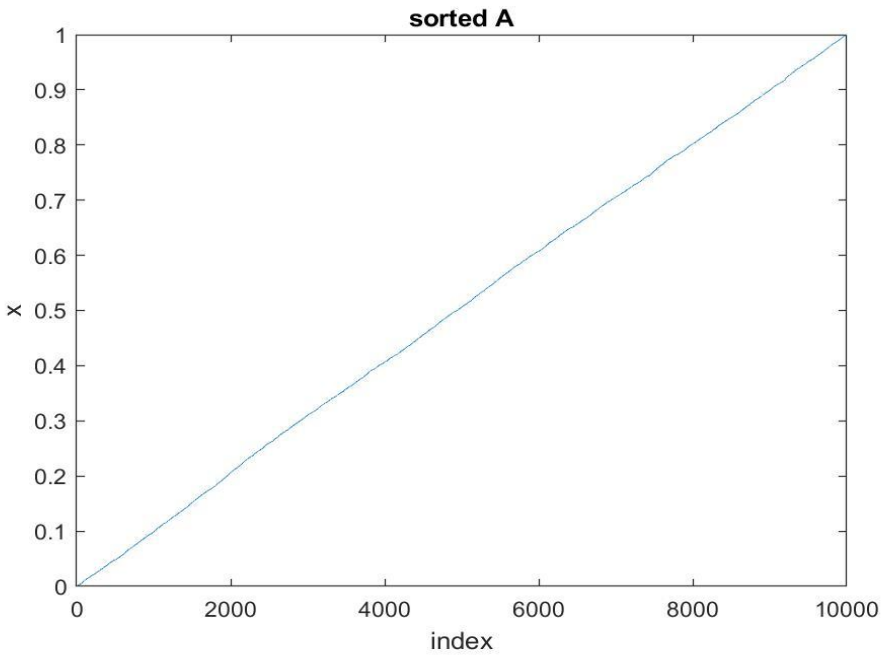
```
x=-10:10;
y=1./(1+exp(-x));
plot(x,y)
title('sigmoid function');
xlabel('x');
ylabel('y');
```



1.4. Given a matrix A=rand(100,100), write a few lines of code to do each of the following. Try to avoid using loops.

a. Sort all the elements in A, put the result in a single 10,000-dimensional vector x, and plot the values in x

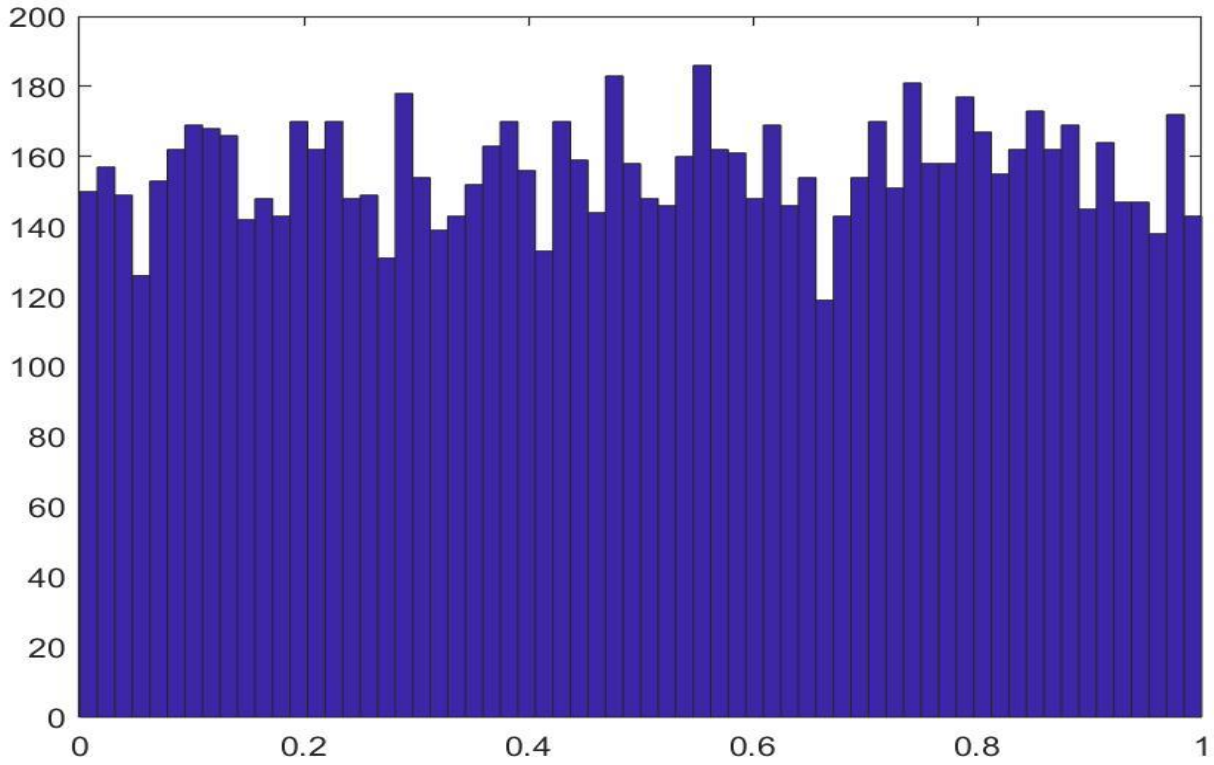
```
A=rand (100,100);
x = sort(A(:));
plot(x);
xlabel('index');
ylabel('x');
title('sorted A')
```



- **Note**: Matrix A=rand (100,100) has been given for the following items

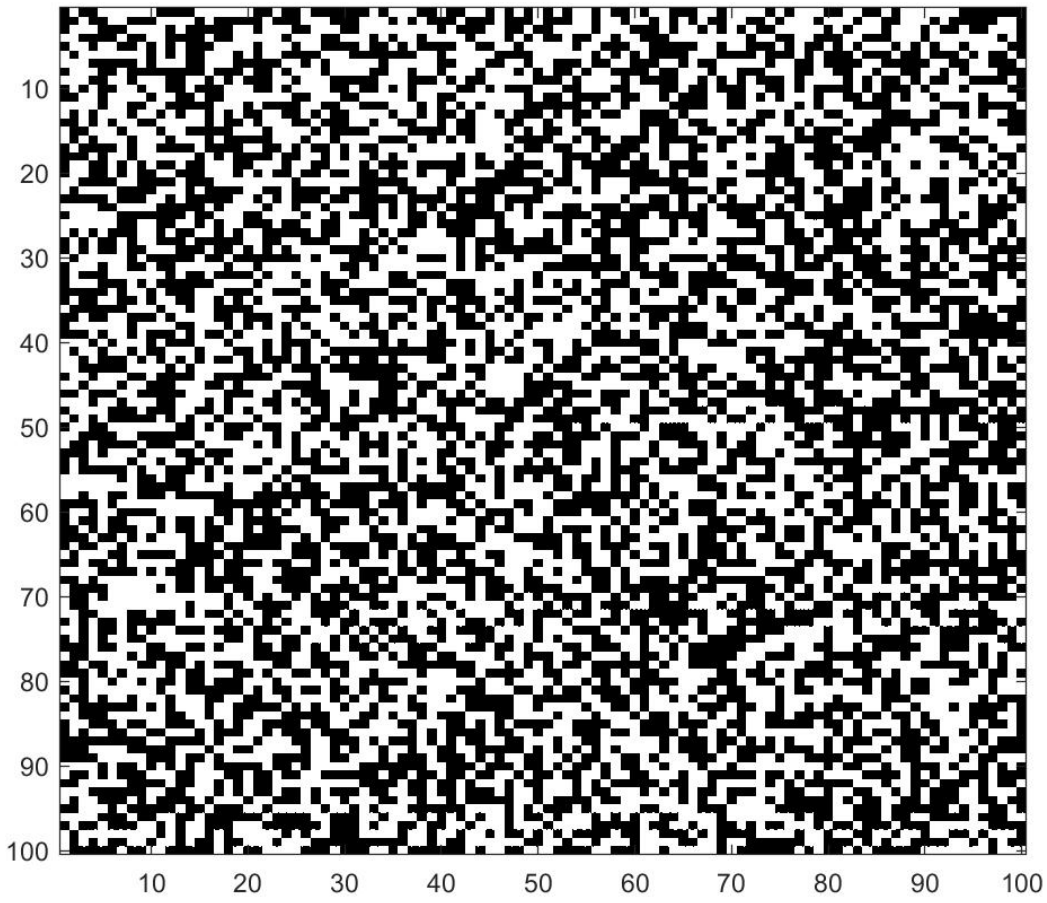
b. Create a 64-bin histogram bar chart of the elements in A. Use hist();

```
hist(A(:),64)
```



c. Create a new matrix with the same size as A, which is 255 wherever the element in A is greater than a threshold t (e.g., 0.5), and 0 everywhere else; call imagesc() to visualize this matrix as an grayscale image;

```
B=zeros(100,100); %Create a 100x100 matrix of zeros
ind=find(A>0.5);% Find indices of elements of A which are greater than
0.5
B(ind)=255; % Set the elements to 255
imagesc(B)
colormap(gray)
```



d. Create a matrix to store the elements in the bottom right quadrant of A.

```
S = size(A) / 2;
A22 = A(S(1)+1:end, S(2)+1:end) % Store the elements of the
bottom right quadrant of matrix in A22
```

e. Create a new matrix that is a duplicate of A; Subtract A's mean value from every element of B; Set any negative element in B to be 0.

```
B=A; Create a duplicate of matrix A
B=B-mean(A); % Subtract A's mean value from every element of B
indexNegative=find(B<0); % Find indices of negative elements
B(indexNegative)=0; % Set the negative elements to zero
```

f. Create a new matrix to include all rows of A whose first column is larger than 0.5 and second column is smaller than 0.8;

```
indexRow=find(A(:,1)>0.5 & A(:,2)<0.8);% Find the rows of A whose
first column is larger than 0.5 and second column is smaller than
0.8
B=A(indexRow,:);
```

g. Create a new matrix by randomly selecting 20 rows from A.

```
n=20; % numbers of rows needed
randomRow=randperm(length(A),n); % create a vector containing 20
unique integers selected randomly from 1 to 100
B=A(randomRow,:)  % new matrix that the rows randomly selected
from A
```

1.5 . Use the function rand () to write a function that returns the roll of a six-sided die.

```
function [side] = roll_fair_six_sided_dice
% rolls a fair six-sided dice and returns 1,2,3,4,5,6 with equal
probability

p = rand; % create random number in the interval (0,1)

if (p < 1/6)
    side = 1;
elseif (p < 2/6)
    side = 2;
elseif (p < 3/6)
    side = 3;
elseif (p < 4/6)
    side = 4;
elseif (p < 5/6)
    side = 5;
else
    side = 6;
end
```