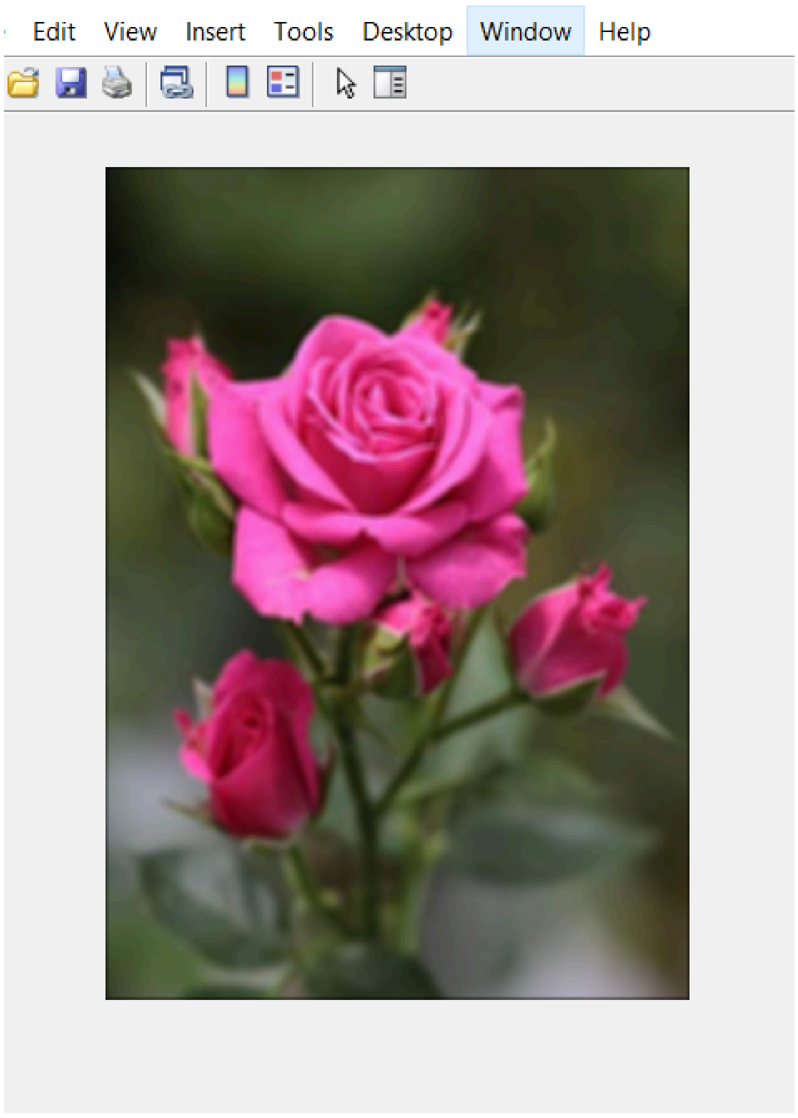
TASK : 1

- As the filter size increases,the blur in the output image is stronger. Larger filter sizes may cause greater smoothing and loss of fine details in the image
- The pixels near the border of the image are affected differently because convolution involves sliding the filter over the image. For pixels near the edges, the filter extends beyond the image, causing the border pixels to become darker. This is because there are less surrounding pixels to take an average from

CODE:

```
clc
clear all
close all
%Read image, visualize it, and display its size and class.
I = imread("flower.jpg");
% Convert to double precision
I = im2double(I);
% Display
figure,
imshow(I)
whos I
% Split the image into the 3 component color intensity planes or matrices.
[R, G, B] = imsplit(I);
whos R
% Create a 2-D convolution filter or kernel for averaging or smoothing
W = 5;
h = ones(W)./W.^2
%Convolve R with h to blur the image
Ravg = conv2(R,h,"same");
% Convolve G with h to blur the image
Gavg =  conv2(G,h,"same");
% Convolve B with h to blur the image
Bavg = conv2(B,h,"same");
% Concatenate the three matrices to put together the color image
Iavg = cat(3,Ravg,Gavg,Bavg);
whos Iavg
figure
imshow(Iavg,[])
```

SNAPSHOT:



TASK : 2

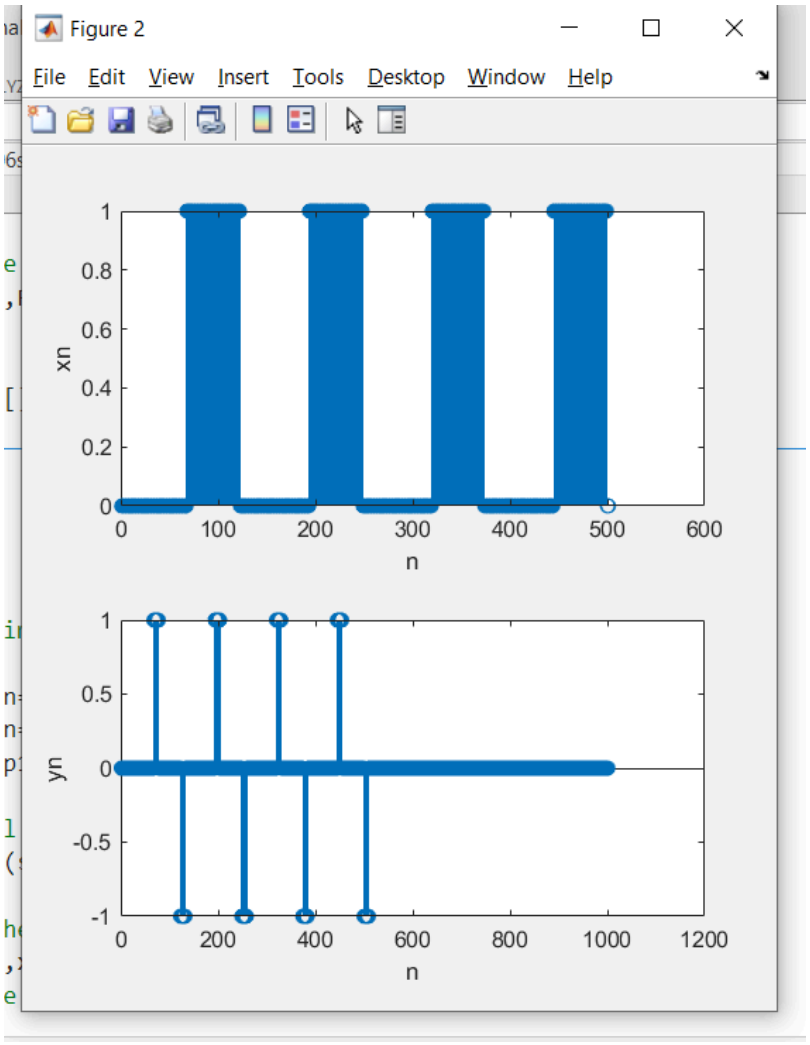
- The transition points in the output signal can be observed to be the same as in the input signal. This shows that the first difference filter can detect the transition points in a matrix(or edges in an image). The rising points of the input are represented by positive impulses and falling edges by negative impulses.

CODE:

```
clc
clear all
%generating impulse response by sutracting two impulses
n=0:0.1:50;
unitstep1 = n==0;
unitstep2 = n==1;
hn = unitstep1 - unitstep2; %impulse response
%input signal
xn = double((sin(0.5*n)+0.2)< 0);
%covolving the input with impulse response
```

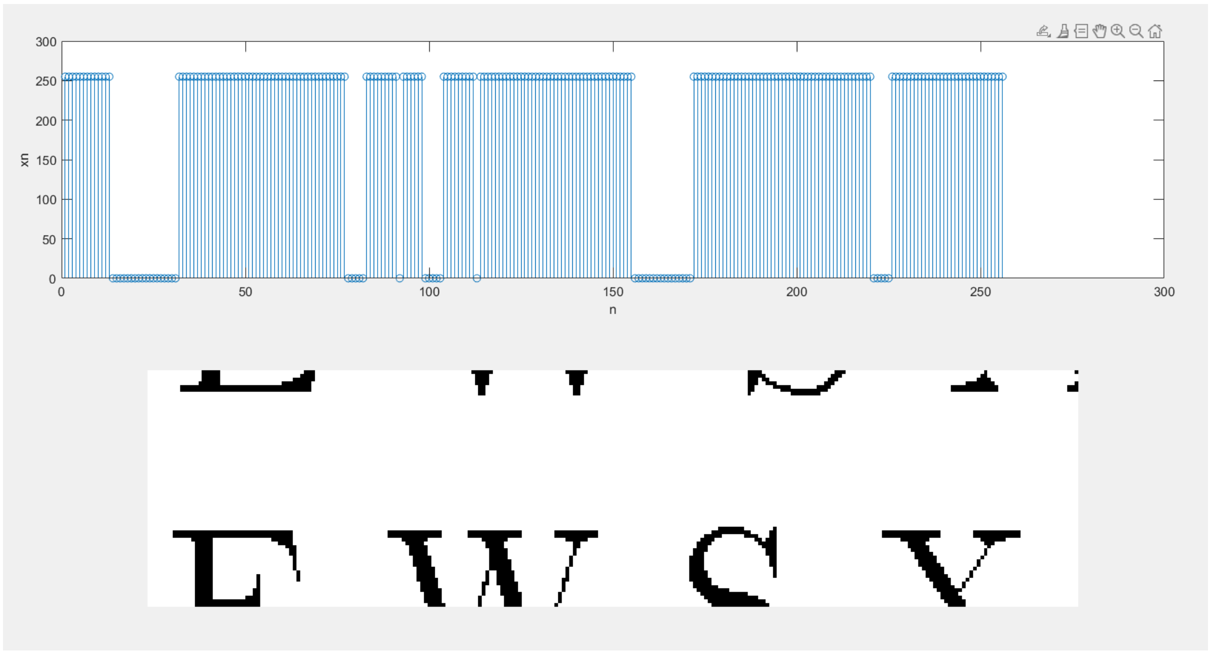
```
yn = conv(hn,xn);
%plotting the input and output
subplot 211
stem(xn)
xlabel('n')
ylabel('xn')
subplot 212
stem(yn)
xlabel('n')
ylabel('yn')
```

SNAPSHOT:

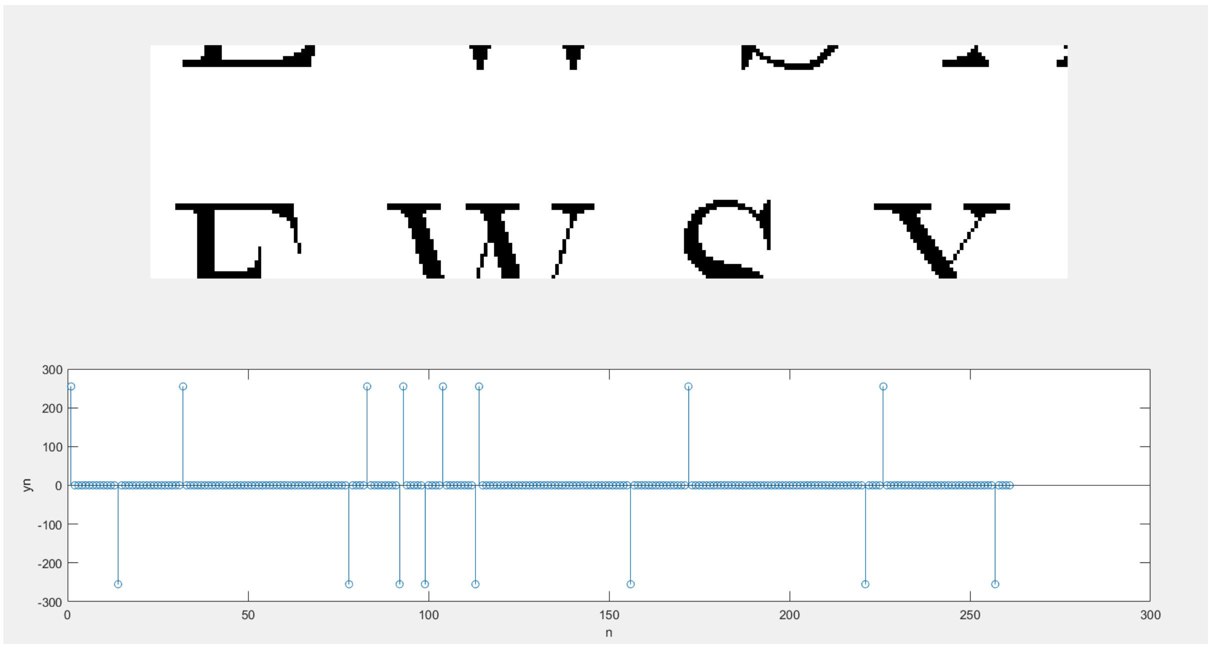


TASK : 3

- In the input signal the zeros represent the black pixels and the ones represent the black pixels as seen here:



- In the output, the nonzero values represent transition between black and white pixels. -1s represent transition from white to black, 1s represent transition from black to white as seen below:

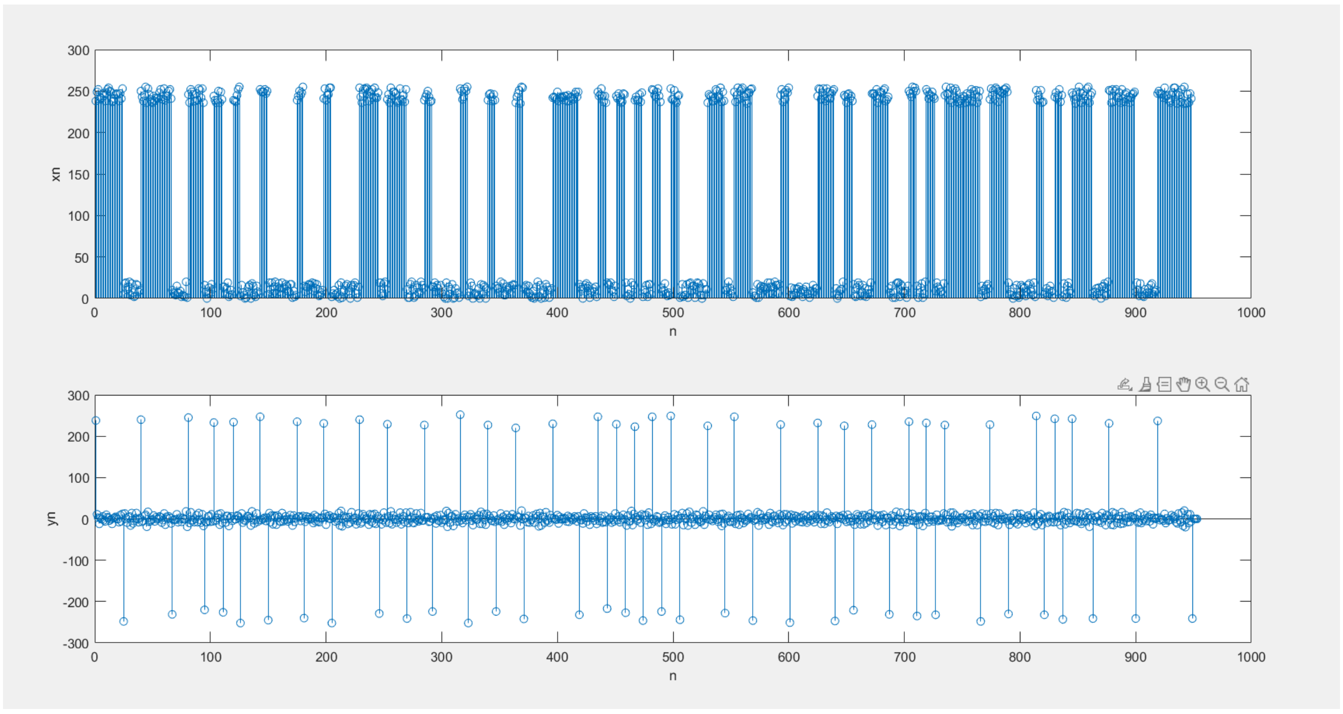


CODE:
```
clc
close all
clear all
%generating impulse response of first difference by sutracting two impulses
n=0:1:5;
unitstep1 = n==0;
unitstep2 = n==1;
```

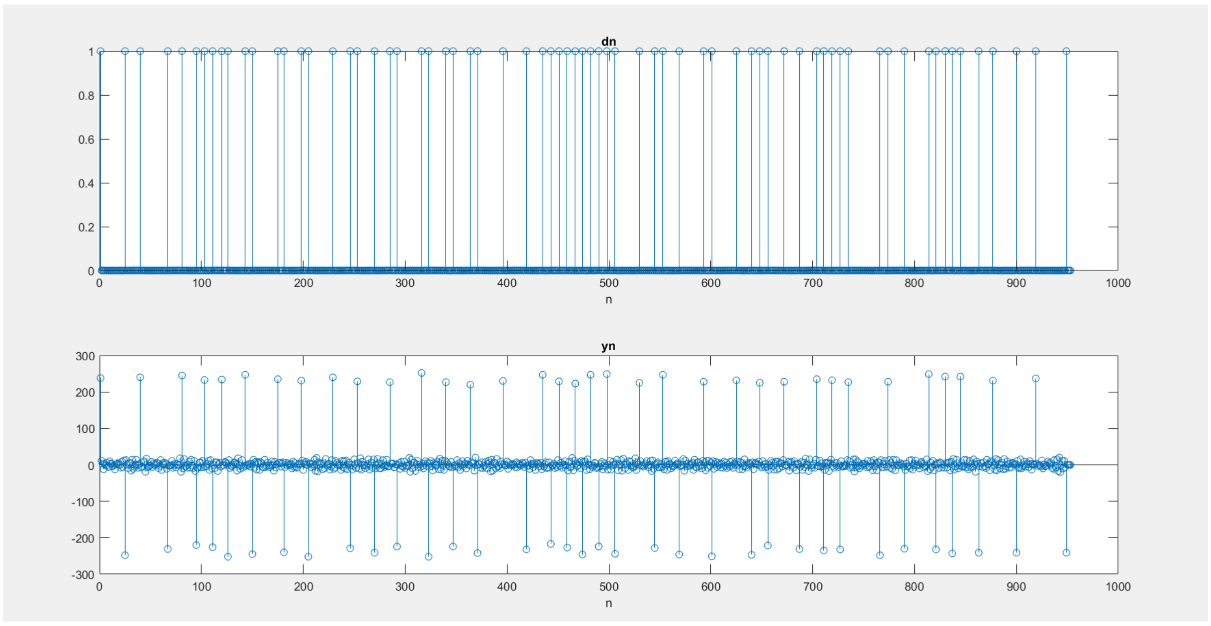
```
hn = unitstep1 - unitstep2; %impulse response
%load the image
load("echart.mat");
%display all the columns of rows 1 till 65
imshow(echart(1:65,:))
%extract row 65
xn = echart(65,:);
%convolution
yn = conv(hn,xn);
size(yn)
%plotting the input and output
subplot 211
stem(yn)
xlabel('n')
ylabel('xn')
subplot 212
stem(yn)
xlabel('n')
ylabel('yn')
```

TASK : 4

- The difference in magnitudes is due to the difference in the intensity of the colour.
- The find function takes a vector and returns another vector that contains all the indices where a nonzero element occurs in the input vector.



- Plot of dn and yn:



CODE:

```
clc
clear all
close all
%generating impulse response of first difference by sutracting two impulses
n=0:1:5;
unitstep1 = n==0;
unitstep2 = n==1;
hn = unitstep1 - unitstep2; %impulse response
%read the barcode image
I = imread("HP110v3.png");
% figure
% imshow(I);
%extract one row
xn = I(65,:);
%convolution
yn = conv(hn,xn);
%plot of xn and yn together in continuous
% subplot 211
% plot(yn)
% ylabel('yn')
% subplot 212
% plot(xn)
% ylabel('yn')
%plotting the xn and yn in discrete
% figure,
% subplot 211
% stem(xn)
% xlabel('n')
% ylabel('xn')
% subplot 212
% stem(yn)
% xlabel('n')
% ylabel('yn')
```

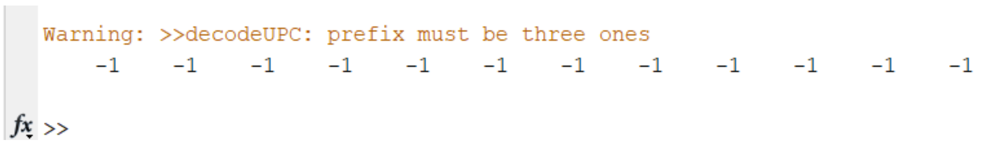
```
thresh = 150; %threshold value
dn = (abs(yn) >= thresh);
%plot of dn and yn together
figure,
subplot 211
stem(dn)
xlabel('n')
title('dn')
subplot 212
stem(yn)
xlabel('n')
title('yn')
location = find(dn); %locations of nonzeros
delta = diff(location);
```

TASK 05

CODE:

```
clc
%width estimation
exp_bars = 59;
width = 95*exp_bars;
theta = width/sum(delta)
fixed_delta = round(delta / estimated_theta);
% obtaining 59 bars
start_index = 4 % Adjust this value based on observation
end_index = start_index + exp_bars - 1
% Check if the indices are within the valid range
if end_index > length(fixed_delta)
end_index = length(fixed_delta);
end
% appropriate section of fixed_delta
delta_section = fixed_delta(start_index:end_index);
% using UPC function
decoded_barcode = decodeUPC(delta_section);
disp(decoded_barcode);
```

SNAPSHOT:



```
Warning: >>decodeUPC: prefix must be three ones
    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1

>>
```