3.2

```
%% 3.2
clc;clear;close all
load echart.mat

bdiffh = [1, -1];

imshow(echart)
m = 65; % 147, 221
yy1 = conv( echart(m,:), bdiffh );

%% Plot the input and output in the same figure using subplot
nn = 1:length(echart(m,:));

subplot(2, 1, 1);
plot(nn, echart(m,:)); grid on;
title('Input Signal');

subplot(2, 1, 2);
plot([nn, length(yy1)], yy1); grid on;
title('Convolved Signal');

%% Find the width of "E"

indices = find(yy1);

%ignore the first impulse

width = indices(3) - indices(2);
disp(width)
```

3.3.2

```
%% 3.3.2
clc;clear;close all

%% Part a
xx = 255*(rem(1:159,30)>19);
bb = [1, -1];
yy = firfilt(bb, xx);

% Plot x and y using subplot
nn1 = 1:length(xx);
subplot(3, 1, 1);
stem(nn1-1, xx(nn1));

title("X Vector");

nn2 = 1:length(yy);
subplot(3, 1, 2);
stem(nn2-1, yy(nn2), 'filled');
title("Y Vector");

%% Part b
% Explain the effect of the first-difference operator on this input signal.
% puts an impulse (positive) when the input signal transitions from 0 to
% HIGH and then gives a negative impulse when the input signal
% transitions from high to low
%% Part c
% Find length of xx and yy
lengthY = length(xx)+length(bb)-1;

%% Part d: find the edges
threshold = 255;
d = abs(yy)>=threshold;

%% Part e: find edges indices
edge_index = find(d)
num_edges = length(edge_index);
rangeIdx = 1:num_edges;

subplot(3, 1, 3);
stem(rangeIdx-1, edge_index(rangeIdx)); grid on;
title("edges in signal");
```

3.3.4

```
%% 3.3.4 (UPC)
clc;clear;close all
img = imread('HP110v3.png');
numPlots = 4;
% img = imread('OFFv3.png');    % Uncomment fo part j

% Take one row in the middle
xx = img( round(size(img, 1)/2) , : );

xxLength = 1:length(xx);
% Apply first difference filter
bb = [1, -1];
yy = firfilt(bb, xx);
yyLength = 1:length(yy);
% Plot input and output using subplot
subplot(numPlots, 1, 1);
stem(xxLength-1, xx(xxLength)); grid on;
title('Input UPC Image');

subplot(numPlots, 1, 2);
stem(yyLength-1, yy(yyLength)); grid on;
title('First Difference Applied to UPC Image');

% Find d[n] and l[n]
threshold = 200 ;
dd = abs(yy) >= threshold;
ll = find(dd);



num_edges = length(ll);
lLength = 1:num_edges;

subplot(numPlots, 1, 3);
stem(lLength - 1, ll(lLength)); grid on;
title('Locations of edges')


% Apply first difference filter to calculate bar widths
delta = firfilt(bb, ll)
lDelta = 1:length(delta);
% Plot l[n] and delta[n] using subplot
subplot(numPlots, 1, 4);

stem(lDelta-1, delta(lDelta)); grid on;
title('First Difference Applied to Location Vector');

% Part e
%  prove that the total width of a valid 12-digit bar code is equal to 95θ
%since there are 12 digits in a UPC, and each digit is encoded with bar
%widths of different orders all totaling 7, there are going to be 84 total
%bar width variations (12*7). But each UPC is delimited by 1-1-1 on each
%end adding 6 bar widths and then is separated in the middle by 1-1-1-1-1
%adding the last 5 to total 95 times the unit bar width.
```

```
% Loop through all the subsets
for start_idx = 1:length(delta)-58+1
    % Take subset of length 59 starting with start_idx
    subset = delta(start_idx:start_idx+59-1);

    % Part f
    sorted_delta = sort(subset); %sort the delta signal
    num_smallest = 31;  %grab the smallest values
    theta = median(sorted_delta(1:num_smallest))+1; %take the average of
those small values
    % theta = 6;

    % Part g
    width_arr = round(subset / theta); %divide the delta signal by the
average width and then round

    % Part h (decodeUPC.p is provided)
    code = decodeUPC(width_arr);

    % Check for incorrect codes
    incorrect = any(code == -1); %if any element in codes is -1, incorrect is
true
    % Continue for loop if incorrect; break out if correct
    if (~incorrect)
        break;
    end

end

%% Printing Detected code
code
```