Student Name: Emirhan Uçar (25265), Emre Eren (25139)

Date of Submission: 17.04.2020

EE312 LAB REPORT#3
SPRING 2020

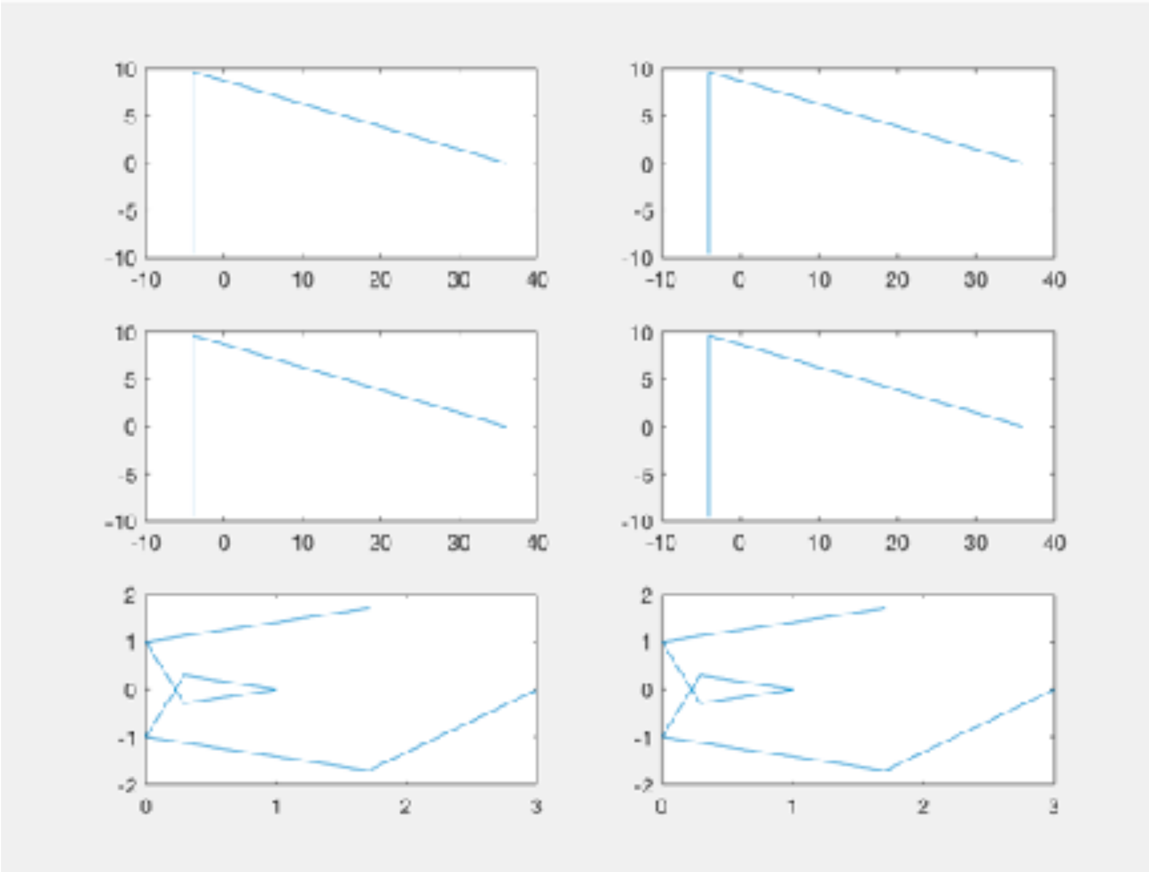
Problem 3.1

a) x1=[1:8];

b) x2=ones(1,8);

c) x3=[1 1 1 0 0 0 0 0];

```
K = dftmtx(8);
x1 = [1:8]';
x2 = ones(1,8)';
x3 = [1 1 1 0 0 0 0 0]';

y1_k = K*x1;
y1 = fft(x1);
subplot(3,2,1); plot(y1_k);
subplot(3,2,2); plot(y1);

y2_k = K*x2;
y2 = fft(x2);
subplot(3,2,3); plot(y1_k);
subplot(3,2,4); plot(y1);

y3_k = K*x3;
y3 = fft(x3);
subplot(3,2,5); plot(y3_k);
subplot(3,2,6); plot(y3);
```



Both of the processes are same, as can be seen in graph we obtain exactly same results. As we know DFT is matrix multiplication with n*n matrix and, since DFT and FFT has same task, we obtain same result. Left hand side graphes represent matrix multiplication and right hand side graphes represent fft.

Problem 3.2

Firstly, we assigned the parameter to our function N which is the length of the input. According to our test, behavior of original dftmtx function and function that we designed are same.

```
function D = mydftmtx(N)
    template = zeros(N,N);
    wn = exp(-1i*2*pi*(1/N));


    for i=1:1:N
        for j=1:1:N
            template(i,j) = wn^((i-1)*(j-1));
        end
    end

    D = template;

end
```

We wrote our function in one script. Then, we assigned the output of the function to variable as "D = mydftmtx(16);". Then we gave an input to our function as length N = 16 from command window. After that, we created two variable as D1 and D2 to compare the results of the matrix multiplication of D and its transpose with respect to commutative property.

$$D_N = \begin{array}{c} \\ k=0 \\ k=1 \\ \vdots \\ k=N-1 \end{array} \overset{n=0 \quad n=1 \ \cdots \ n=N-1}{\begin{bmatrix} & & \\ & e^{-j\frac{2\pi k}{N}n} & \\ & & \end{bmatrix}}_{N\times N} \rightarrow \text{the entry for row } k \text{ and column } n$$

$$\downarrow W_N^{kn} \qquad \left(\text{remember } W_N = e^{-j2\pi/N}\right)$$

$$D_N = \begin{bmatrix} W_N^{0\cdot0} & W_N^{0\cdot1} & \cdots & W_N^{0\cdot(N-1)} \\ W_N^{1\cdot0} & W_N^{1\cdot1} & \cdots & W_N^{1\cdot(N-1)} \\ \vdots & \vdots & & \vdots \\ W_N^{(N-1)\cdot0} & W_N^{(N-1)\cdot1} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix}, \text{ so } (D_N)_{kn} = W_N^{n\cdot k} \quad \begin{pmatrix} k=0\ldots N-1 \\ n=0\ldots N-1 \end{pmatrix}$$

D1 = (D*(D'));
D2 = ((D')*D);

and we got tables at the below. We inferred from the table that diagonals of both matrix are same.

D2

16x16 complex double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	16.0000 ...	1.9984e-...	2.2204e-...	2.1094e-...	1.7764e-...	1.7208e-...	1.1102e-...	4.4409e-...	0.0000e...	-5.5511e...	-1.7764e...	-3.4972e...	-5.3291e...	-8.4377e...	-1.5099e...	-3.1085e...
2	1.9984e-...	16.0000 ...	2.3574e-...	2.2480e-...	2.0779e-...	2.0113e-...	1.4486e-...	7.0895e-...	5.9566e-...	5.4508e-...	-1.1062e...	-1.7125e...	-3.4027e...	-5.0305e...	-8.3737e...	-1.5987e...
3	2.2204e-...	2.3574e-...	16.0000 ...	2.1622e-...	1.9538e-...	1.7675e-...	2.2885e-...	1.4062e-...	1.0275e-...	4.3901e-...	3.5336e-...	-7.9529e...	-2.0865e...	-3.1332e...	-5.3291e...	-8.1010e...
4	2.1094e-...	2.2480e-...	2.1622e-...	16.0000 ...	1.9407e-...	1.5526e-...	2.2539e-...	1.7893e-...	1.4857e-...	1.2887e-...	6.9444e-...	4.1640e-...	-8.2617e...	-1.7764e...	-3.4614e...	-5.3635e...
5	1.7764e ...	2.0779e ...	1.9538e ...	1.9407e ...	16.0000 ...	2.2781e ...	2.3761e ...	1.7522e ...	1.7764e ...	1.1972e ...	1.0438e ...	3.9074e ...	0.0000e...	-9.4585e...	-1.7100e...	-3.3622e...
6	1.7208e ...	2.0113e ...	1.7675e ...	1.5526e ...	2.2781e ...	16.0000 ...	2.8281e ...	2.0815e ...	2.0419e ...	2.0071e ...	1.8824e ...	-8.8818e...	6.0413e ...	5.5975e ...	-5.5624e...	-1.5659e...
7	1.1102e-...	1.4486e-...	2.2885e-...	2.2539e-...	2.3761e-...	2.8281e-...	16.0000 ...	2.1357e-...	2.1377e-...	1.3604e-...	0.0000e...	1.8318e-...	9.7532e-...	8.0105e-...	-3.5218e...	-8.7978e...
8	4.4409e-...	7.0895e-...	1.4062e-...	1.7893e-...	1.7522e-...	2.0815e-...	2.1357e-...	16.0000 ...	2.3176e-...	2.0645e-...	1.4110e-...	1.6740e-...	1.5709e-...	1.3442e-...	7.7545e-...	2.8071e-...
9	0.0000e...	5.9566e-...	1.0275e-...	1.4857e-...	1.7764e-...	2.0419e-...	2.1377e-...	2.3176e-...	16.0000 ...	1.6792e-...	1.9701e-...	2.0660e-...	1.7764e-...	1.8230e-...	1.2819e-...	6.2559e-...
10	-5.5511e...	5.4508e-...	4.3901e-...	1.2887e-...	1.1972e-...	2.0071e-...	1.3604e-...	2.0645e-...	1.6792e-...	16.0000 ...	1.5850e-...	2.5256e-...	1.7929e-...	1.9003e-...	1.9565e-...	3.7588e-...
11	-1.7764e...	-1.1062e...	3.5336e-...	6.9444e-...	1.0438e-...	1.8824e-...	0.0000e...	1.4110e-...	1.9701e-...	1.5850e-...	16.0000 ...	2.4906e-...	2.3086e-...	1.6482e-...	1.8444e-...	1.0093e-...
12	-3.4972e...	-1.7125e...	-7.9529e...	4.1640e-...	3.9074e-...	-8.8818e...	1.8318e-...	1.5740e-...	2.0660e-...	2.5256e-...	2.4906e-...	16.0000 ...	2.6025e-...	1.4596e-...	2.0957e-...	1.9558e-...
13	5.3291e...	3.4027e...	2.0865e...	8.2617e...	0.0000e...	6.0413e ...	9.7532e ...	1.5709e ...	1.7764e ...	1.7929e ...	2.3086e ...	2.6025e ...	16.0000 ...	1.6163e ...	2.0213e ...	1.0263e ...
14	-8.4377e...	-5.0305e...	-3.1332e...	-1.7764e...	-9.4585e...	5.5975e-...	8.0105e-...	1.3442e-...	1.5230e-...	1.9003e-...	1.5482e-...	1.4596e-...	1.5163e-...	16.0000 ...	2.2776e-...	2.2480e-...
15	-1.5099e...	-8.3737e...	-5.3291e...	-3.4614e...	-1.7100e...	-5.5624e...	-3.5218e...	7.7545e-...	1.0819e-...	1.9565e-...	1.8444e-...	2.0957e-...	2.0213e-...	2.2776e-...	16.0000 ...	2.0199e-...
16	-3.1085e...	-1.5987e...	-8.1010e...	-5.3635e...	-3.3622e...	-1.5659e...	-8.7978e...	2.8071e-...	6.2559e-...	3.7588e-...	1.0093e-...	1.9558e-...	1.9263e-...	2.2480e-...	2.2899e-...	16.0000 ...

D1

16x16 complex double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	16.0000 ...	1.9984e-...	2.2204e-...	2.1094e-...	1.7764e-...	1.7208e-...	1.1102e-...	4.4409e-...	0.0000e...	-5.5511e...	-1.7764e...	-3.4972e...	-5.3291e...	-8.4377e...	-1.5099e...	-3.1085e...
2	1.9984e-...	16.0000 ...	2.3574e-...	2.2480e-...	2.0779e-...	2.0113e-...	1.4486e-...	7.0895e-...	5.9566e-...	5.4508e-...	-1.1062e...	-1.7125e...	-3.4027e...	-5.0305e...	-8.3737e...	-1.5987e...
3	2.2204e-...	2.3574e-...	16.0000 ...	2.1622e-...	1.9538e-...	1.7675e-...	2.2885e-...	1.4062e-...	1.0275e-...	4.3901e-...	3.5336e-...	-7.9529e...	-2.0865e...	-3.1332e...	-5.3291e...	-8.1010e...
4	2.1094e-...	2.2480e-...	2.1622e-...	16.0000 ...	1.9407e-...	1.5526e-...	2.2539e-...	1.7893e-...	1.4857e-...	1.2887e-...	6.9444e-...	4.1640e-...	-8.2617e...	-1.7764e...	-3.4614e...	-5.3635e...
5	1.7764e-...	2.0779e-...	1.9538e-...	1.9407e-...	16.0000 ...	2.2781e-...	2.3761e-...	1.7522e-...	1.7764e-...	1.1972e-...	1.0438e-...	3.9074e-...	0.0000e...	-9.4585e...	-1.7100e...	-3.3622e...
6	1.7208e-...	2.0113e-...	1.7675e-...	1.5526e-...	2.2781e-...	16.0000 ...	2.8281e-...	2.0815e-...	2.0419e-...	2.0071e-...	1.8824e-...	-8.8818e...	6.0413e-...	5.5975e-...	-5.5624e...	-1.5659e...
7	1.1102e ...	1.4486e ...	2.2885e ...	2.2539e ...	2.3761e ...	2.8281e ...	16.0000 ...	2.1357e ...	2.1377e ...	1.3604e ...	0.0000e...	1.8318e ...	9.7532e ...	8.0105e ...	-3.5218e...	-8.7978e...
8	4.4409e ...	7.0895e ...	1.4062e ...	1.7893e ...	1.7522e ...	2.0815e ...	2.1357e ...	16.0000 ...	2.3176e ...	2.0645e ...	1.4110e ...	1.5740e ...	1.5705e ...	1.3442e ...	7.7545e ...	2.8071e ...
9	0.0000e...	5.9566e-...	1.0275e-...	1.4857e-...	1.7764e-...	2.0419e-...	2.1377e-...	2.3176e-...	16.0000 ...	1.6792e-...	1.9701e-...	2.0660e-...	1.7764e-...	1.8230e-...	1.0815e-...	6.2559e-...
10	-5.5511e...	5.4508e-...	4.3901e-...	1.2887e-...	1.1972e-...	2.0071e-...	1.3604e-...	2.0645e-...	1.6792e-...	16.0000 ...	1.5850e-...	2.5256e-...	1.7929e-...	1.9003e-...	1.9565e-...	3.7588e-...
11	-1.7764e...	-1.1062e...	3.5336e-...	6.9444e-...	1.0438e-...	1.8824e-...	0.0000e...	1.4110e-...	1.9701e-...	1.5850e-...	16.0000 ...	2.4906e-...	2.3086e-...	1.6482e-...	1.8444e-...	1.0093e-...
12	-3.4972e...	-1.7125e...	-7.9529e...	4.1640e-...	3.9074e-...	-8.8818e...	1.8318e-...	1.6740e-...	2.0660e-...	2.5256e-...	2.4906e-...	16.0000 ...	2.6025e-...	1.4696e-...	2.0952e-...	1.9558e-...
13	-5.3291e...	-3.4027e...	-2.0865e...	-8.2617e...	0.0000e...	6.0413e-...	9.7532e-...	1.5705e-...	1.7764e-...	1.7929e-...	2.3086e-...	2.6025e-...	16.0000 ...	1.6163e-...	2.0213e-...	1.0263e-...
14	-8.4377e...	-5.0305e...	-3.1332e...	-1.7764e...	-9.4585e...	5.5975e-...	8.0105e-...	1.3442e-...	1.5230e-...	1.9003e-...	1.5482e-...	1.4696e-...	1.5163e-...	16.0000 ...	2.2776e-...	2.2480e-...
15	1.5099e...	8.3737e...	5.3291e...	3.4614e...	1.7100e...	5.5624e...	3.5218e...	7.7545e ...	1.0815e ...	1.9565e ...	1.8444e ...	2.0957e ...	2.0213e ...	2.2776e ...	16.0000 ...	2.0199e ...
16	3.1085e...	1.5987e...	8.1010e...	5.3635e...	3.3622e...	1.5659e...	8.7978e...	2.8071e ...	6.2559e ...	3.7588e ...	1.0093e ...	1.9558e ...	1.9263e ...	2.2480e ...	2.0199e ...	16.0000 ...

Problem 3.3

```
y1 = fft(x1);
y2 = fft(x2);
y3 = fft(x3);
```

These are fft's of sequences that given problem 3.1

```
y1_shift = fftshift(y1);
y2_shift = fftshift(y2);
y3_shift = fftshift(y3);
```

In this part we apply fftshift to the three DFTs that we found in Problem 3.1

```
x = [(-100*pi),(-300*pi/4),(-200*pi/4),(-100*pi/4),(100*pi/4),(200*pi/4),(300*pi/4),(100*pi)];
```

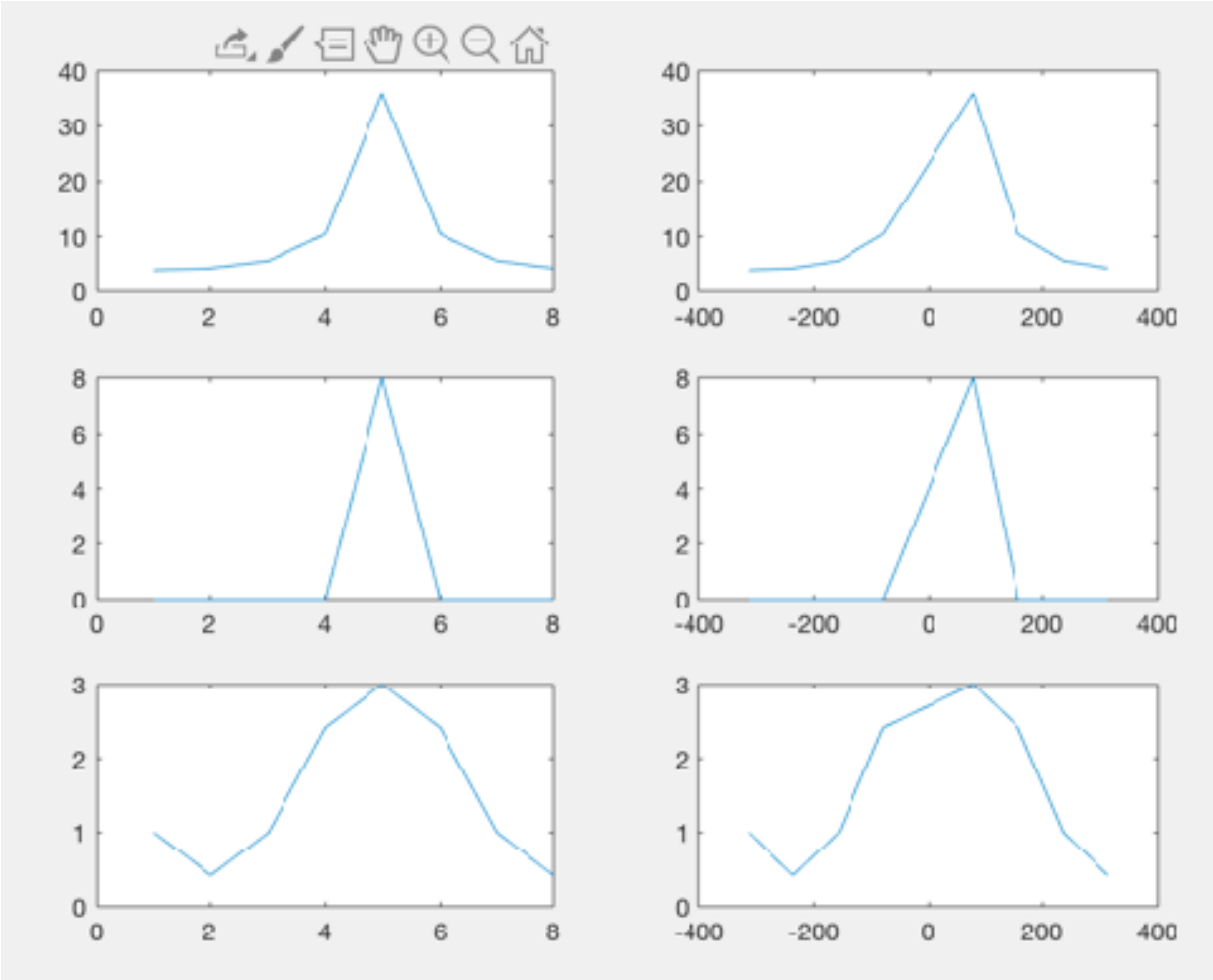
In other words, x can be defined as [-pi : pi/4 : pi]*100

-Range is -pi to pi with interval 2pi/N. Since N = 8, interval is 2pi/8 = pi/4

-100Hz is sampling.

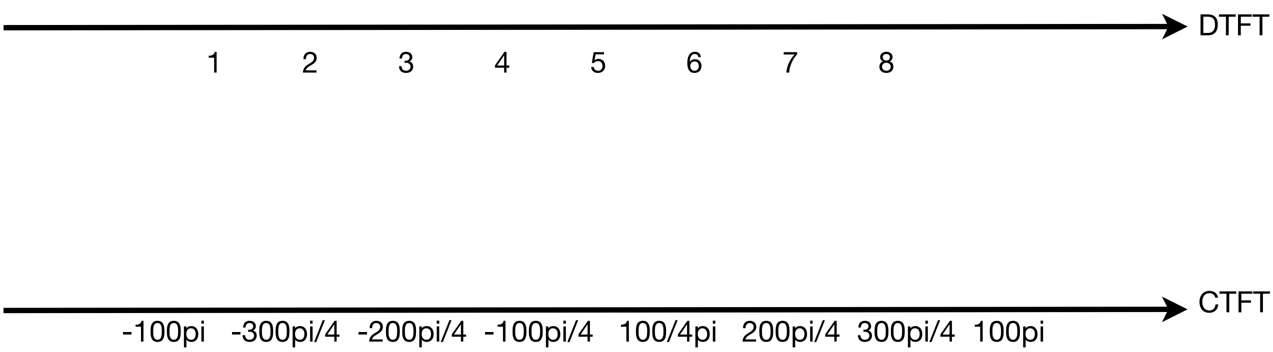
```
subplot(3,2,1); A = plot(abs(y1_shift));
subplot(3,2,2); A_CTFT = plot(x,abs(y1_shift));

subplot(3,2,3); B = plot(abs(y2_shift));
subplot(3,2,4); B_CTFT = plot(x,abs(y2_shift));

subplot(3,2,5); p3=plot(abs(y3_shift));
subplot(3,2,6); p3_CTFT= plot(x,abs(y3_shift));
```

These are applications of magnitudes of fft-shifted DTFTs and their corresponding CTFT



(Left hand side represents DTFT and right hand side represents CTFT)

There is no change in magnitude, only frequency axis is changed as we expected. This conversion can be seen below



Problem 3.4

```
x1 = [1:8]';
x2 = ones(1,8)';
x3 = [1 1 1 0 0 0 0 0]';

myconv1 = mycirconv(x1,x2);
myconv2 = mycirconv(x2,x3);
cc1    = cconv(x1, x2,8);
cc2    = cconv(x2, x3,8);

function out = mycirconv(a,b)
    A = fft(a);
    B = fft(b);
    out = ifft(A.*B);
end
```

x1, x2, x3 are sequences that provided in problem 3.1.

myconv1 and myconv2 are generated by our function and cc1 and cc2 are generated by function that provided by Matlab that called cconv.

This function basically do take fft's of signals, and multiply them in element wise, then take ifft of this multiplication. After that we get required output.

Workspace

Name ▲	Value
cc1	[36;36;36;36;36;36;36;36]
cc2	[3;3;3;3;3;3;3;3]
myconv1	[36;36;36;36;36;36;36;36]
myconv2	[3;3;3;3;3;3;3;3]
x1	[1;2;3;4;5;6;7;8]
x2	[1;1;1;1;1;1;1;1]
x3	[1;1;1;0;0;0;0;0]

As can be seen above our results are matching with Matlab generated circler convolution's results.

Problem 3.5

We created the function named myconv to convolve two inputs linearly by using DFT. Firstly, we added zero padds at the end of the both input. We paid attention to linear convolution length while we are putting zero padds in terms of length of linear convolution is M+N-1. After that we did same process actually in Problem 3.4.

```
function conv = myconv(input1,input2);

    len1 = length(input1);
    len2 = length(input2);

    input1zeropadd = [input1 zeros(1,len2-1)];
    input2zeropadd = [input2 zeros(1,len1-1)];

    input1 = fft(input1zeropadd)
    input2 = fft(input2zeropadd)

    conv =ifft(input1.*input2);

end
```

PART A

Firstly we assigned two inputs to two different variable(a,b) in other script. After that, we wrote myconv(a,b) and conv(a,b) to compare the results of the function that we designed and default function of MATLAB and we got same results thanks to god.

```
ans =

  Columns 1 through 9

     1     2     3     4     5     6     7     7     7

  Columns 10 through 16

     7     6     5     4     3     2     1
```

```
    6.0000    7.0000    7.0000    7.0000    7.0000

  Columns 11 through 15

    6.0000    5.0000    4.0000    3.0000    2.0000

  Column 16

    1.0000
```

PART B

Firstly we assigned ones(1,5) to a variable in other script. After that, we wrote myconv(x1,b) and conv(x1,b) to compare the results of the function that we designed and default function of MATLAB and we got same results.

```
ans =

  Columns 1 through 9

     1     3     6    10    15    20    25    30    26

  Columns 10 through 12

    21    15     8

ans =

  Columns 1 through 9

     1     3     6    10    15    20    25    30    26

  Columns 10 through 12

    21    15     8
```

Problem 3.6

```
[x,fs]= audioread('music1.wav');

h = [0.2, 0.2, 0.2, 0.2, 0.2];

y = fftfilt(h,x);
ee = myfftfilt(x,h);

%sound(ee)
sound(y);

function y = myfftfilt(x,h)
    x = x.'; %converting column vector
    len_x = length(x);
    len_h = length(h);
    total = len_x + len_h - 1;
    h = [h zeros(1,total-len_h)]
    x = [x zeros(1,total-len_x)]
    H = fft(h);
    X = fft(x);
    for i =1:total
        Y(i) = X(i).*H(i);
    end
    y = ifft(Y);
end
```

This function firstly do zero pad to x and h to make both as an equal length, then compute their fft's. After that, we generate Y (which, will be fft of our result) with element wise multiplication of X and H. With this method, actually we are able to obtain y = x*h, because y = x*h means Y = X.*H in frequency domain, therefor we just need to convert Y to y. Hence, we just need to take inverse of Y which is y = ifft(Y).

We cannot detect any differences between myfftfilt(x,h) and fftfilt(h,x), so we believe that our function is perfectly work. Following part can be evaluated as our proof.

y x ee

1x234535 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	-7.0801e...	-0.0051	-0.0123	-0.0171	-0.0167	-0.0114	-0.0029	0.0041	0.0060	-0.0014	-0.0169	-0.0295	-0.0352

ee y

234531x1 double

	1
1	-7.0801e...
2	-0.0051
3	-0.0123
4	-0.0171
5	-0.0167
6	-0.0114
7	-0.0029
8	0.0041
9	0.0060
10	-0.0014
11	-0.0169
12	-0.0295
13	-0.0352

As can be seen in code part, ee = myfftfilt(x,h) and h = fftfilt(h,x). If we make comparison between each elements of ee and h, we can see these two differently generated sequences are same each other. There is only very tiny difference that related with length of the sequence. Since this difference is very small, we can ignore it.