CMPE 362 HW3
Abdurrahim ESKİN
2016400387

Problem 1:

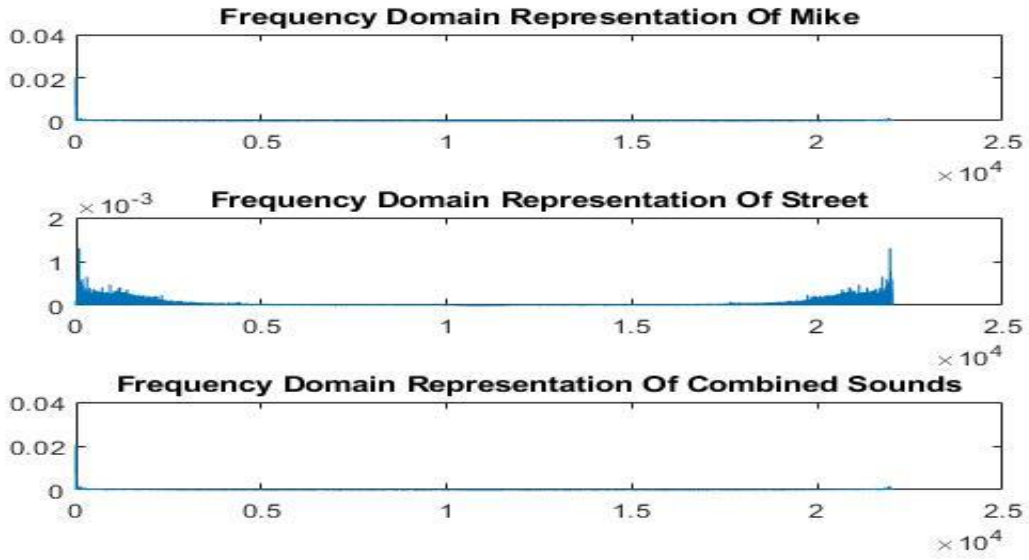
The first figure is the Frequency domain representations of Mike, Street and signal of their combination(one sided).

I simply plot frequency domain representations of Mike.wav, Street.wav, Mike+Street.wav.

These particular plots can be deceive due to scaling. (Attention to y-axis')At first, I simply read the data, take the FFT's to use later in frequency domains.

```
mikeFile = 'mike.wav';
streetFile = 'street.wav';

%y1 for mike, y2 for street, y3 for combined
[y1,Fs1] = audioread(mikeFile);
[y2,Fs2] = audioread(streetFile);
%combined sound of Mike and street
y3 = y1 + y2;

%Lengths of these three sounds
L1 = size(y1,1);
L2 = size(y2,1);
L = size(y3,1);

%Half length
f = Fs1*(0:(L/2))/L;
%Full length
f2 = Fs1*(0:L-1)/L;
f3 = Fs1*(1:L)/L;

%fft of mike
fftMike = fft(y1,L);
p1_full = (fftMike./L);
p1_half = p1_full(1:L/2+1);
p1_half(2:end-1) = 2*p1_half(2:end-1);

%fft of street sound
fftStreet = fft(y2,L);
p2_full = (fftStreet./L);
p2_half = p2_full(1:L/2+1);
p2_half(2:end-1) = 2*p2_half(2:end-1);

% fft of combined sounds
```

```
fftCombined = fft(y3,L);
p3_full = fftCombined./L;
p3_half = p3_full(1:L/2+1);
p3_half(2:end-1) = 2*p3_half(2:end-1);



%Frequency domain representations of mike,street and combined sounds
figure(1)
subplot(3,1,1);
plot(f2,abs(p1_full));
title('Frequency Domain Representation Of Mike')
subplot(3,1,2);
plot(f2,abs(p2_full));
title('Frequency Domain Representation Of Street')
subplot(3,1,3);
plot(f2,abs(p3_full));
title('Frequency Domain Representation Of Combined Sounds')
```

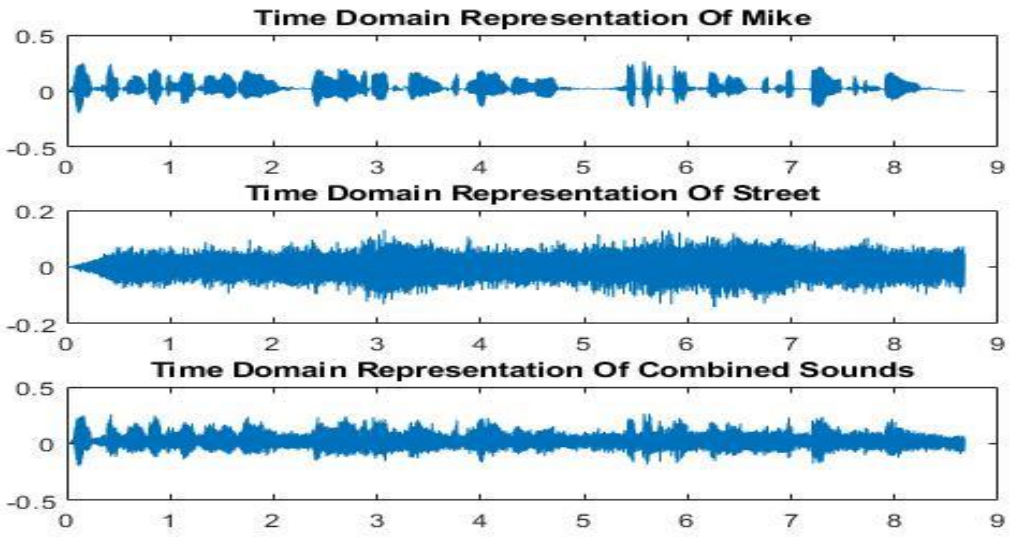


The second figure shows the three signals and their time domain representations

In this part, I simply plot time domain representations of Mike.wav, Street.wav, Mike+Street.wav. Mike's speech is discontinuous which means there are relatively silent moments in his talk.

In street audio, it is continuous. When we combine them together, it is simply adding two audios for every recorded moment. It does not look like either of two before but still it is the combination of two.

```
%Time domain representations of Mike Street and Combined sounds

%Time domain representation of Mike
figure(2)
subplot(3,1,1);
t1=[1:L1]./Fs1;
plot(t1,y1);
title('Time Domain Representation Of Mike')
subplot(3,1,2);
%Time domain representation of Street
t2=[1:L2]./Fs2;
plot(t2,y2);
title('Time Domain Representation Of Street')
subplot(3,1,3);
%Time domain representations of Combiend sounds
t3=[1:L]./Fs1;
plot(t3,y3);
title('Time Domain Representation Of Combined Sounds')
```



In The Filtering Stage:

I found that human voice is between 300 and 3000 Hz.
So, this algorithm simply filter every frequency which is not between 300 and 3000 Hz, symmetrically.

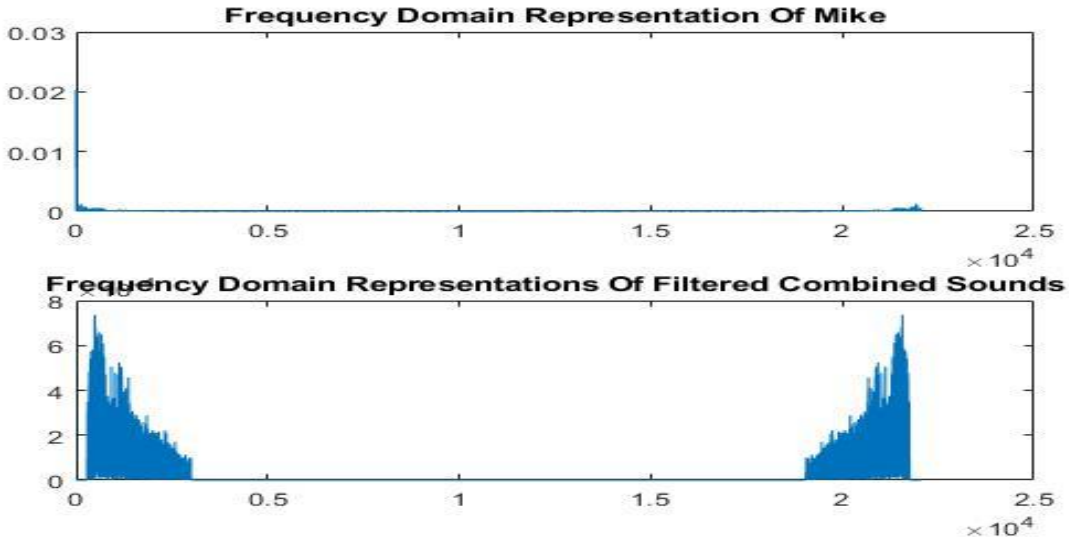
```
%calculations for filtering

p4_half = p3_half; %(half)
p4_full = p3_full; %(full) this will be filtered

%Human voice is between 300-3000 Hz. frequency
```

```
%I am filtering symmetrically
for i = 1:L
    if ~(f3(1,i) > 300 && f3(1,i) < 3000)
        if ~(f3(1,i) < Fs1-300 && f3(1,i) > Fs1-3000 )
            p4_full(i) = 0;
        end
    end
end
```

The third figure shows the frequency domain representations of Mike and filtered combined sounds:

After the calculations, I simply plotted frequency domain representations of Mike.wav, Filtered Mike
+Street.wav. It actually looks somehow similar. Attention to scaling on y-axis. We will test the filtering
system with SNR.

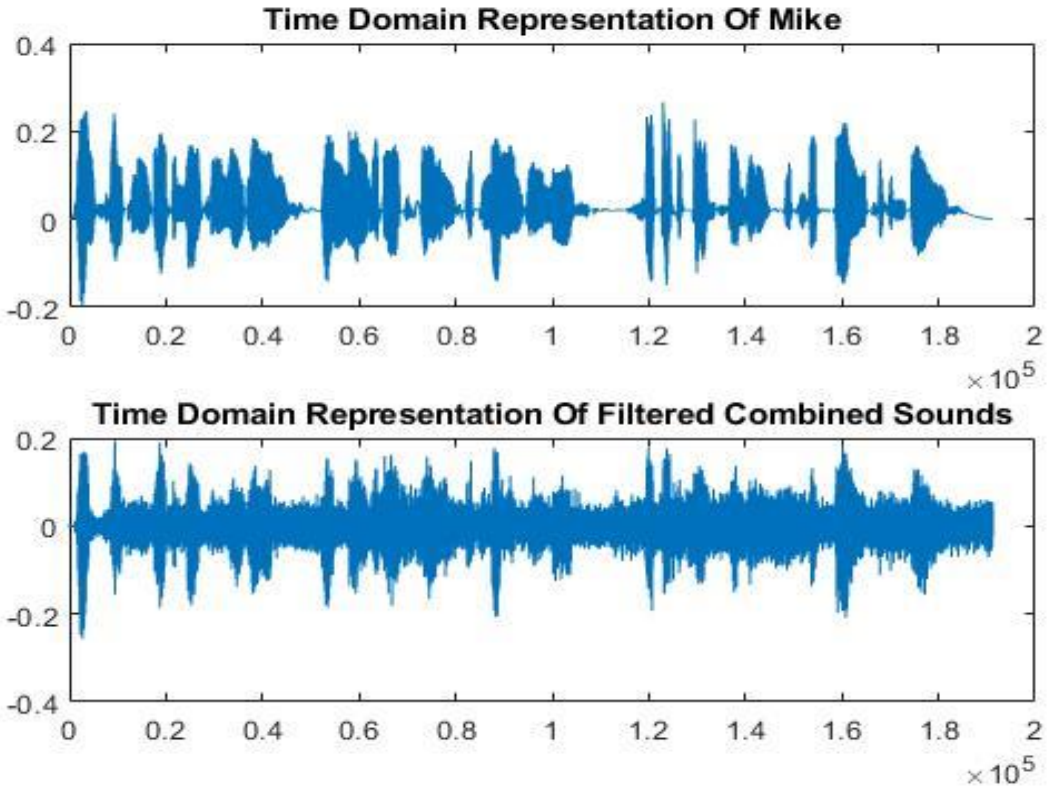
```
%Frequency Domain representation of Mike and filtered combined sounds:

figure(3)
%Frequency Domain representation of Mike
subplot(2,1,1);
plot(f2,abs(p1_full));
title('Frequency Domain Representation Of Mike')

%Frequency Domain representation of filtered combined sounds
subplot(2,1,2);
plot(f3,abs(p4_full));
title('Frequency Domain Representations Of Filtered Combined Sounds')
```



Last figure shows the time domain of Mike and filtered combined sounds:

In this part, I converted the audio from frequency domain to time domain with help of inversed FFT. Then simply plotted time domain representations of Mike.wav, Filtered Mike+Street.wav. We easily can see from the plots that, recovered audio is not the same with the original. Actually there is a relatively big difference between them.

```
%Time domain representations of Mike and filtered combined sounds:
p4_full = p4_full*L;
TimeDomFilt = ifft(p4_full,L);
p1_full = p1_full*L;
TimeDomMike = ifft(p1_full,L);
figure(4)
%Time domain representation Of Mike
subplot(2,1,1);
plot(real(TimeDomMike));
title('Time Domain Representation Of Mike')
%Time domain representation of filtered combined sounds
subplot(2,1,2);
plot(real(TimeDomFilt));
title('Time Domain Representation Of Filtered Combined Sounds')
```



SNR VALUE

To find the error, we can simply use SNR method, it's formula was given in the description. I found SNR as 0.1696. This value for an SNR says that noise is almost as big as the original audio, because SNR value is relatively close to zero.

```
%Calculating the SNR value
y1square = y1.^2;
EiIisquare = (real(TimeDomFilt)-y1).^2;
up = sum(y1square);
down = sum(EiIisquare);
SNR = 10*log(up/down)/log(10)
disp(SNR);
```

Problem 2:

In this question, I used really simple algorithm. At first, I simply read the data, take the FFT's to convert it to frequency domain. For reading different wav files, user should change the first line only. Frequency is high when one claps his hands or snaps his fingers. Claps usually have bigger frequencies than snaps. By trial and error, I found that most of the snaps has lower than 40 Hz. So that, I draw a line between claps and snaps with 40 Hz. If maximum frequency of FFT-taken-data is bigger than 40, it is a clap. Otherwise it is a snap. In other words I just controlled on snap sounds frequency and if frequency bigger than 40 Hz, this sound is a clap.

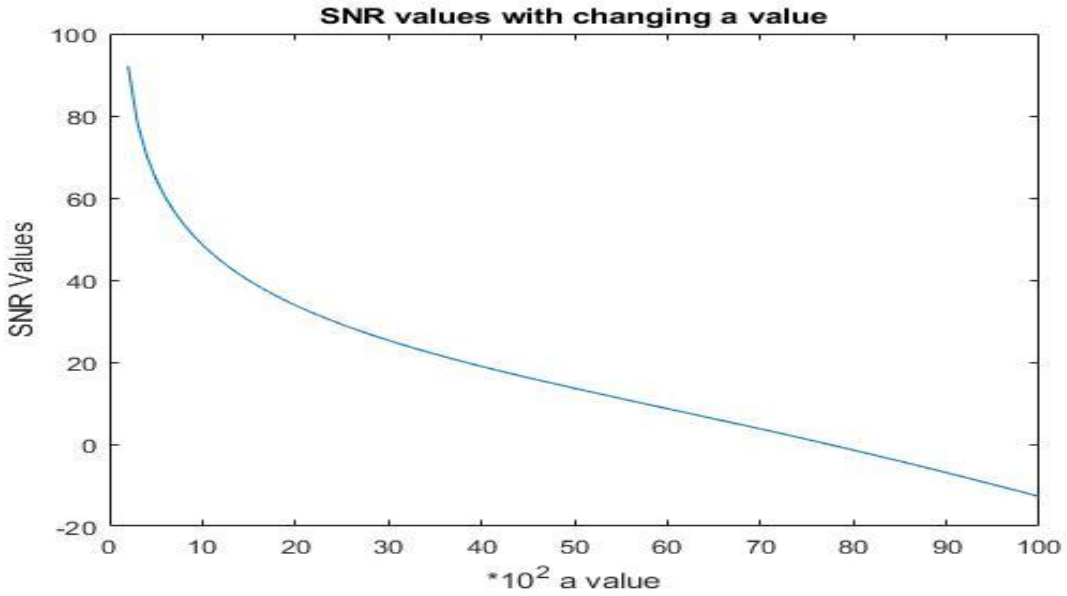
Here is my code for problem 2:

```
hfile = 'snap.wav';

[x, Sound] = audioread(hfile);
Ssize = size(x,1);
%fft of only one column (one ear) of audio
y= fft(x(:,1));

ifSnap = 0;
if abs(max(y))>40 %40 is proper threshold for sound
    ifSnap = 1;
end

if ifSnap == 0
    disp('snap sound detected');
else
    disp('clap sound detected');
end
```

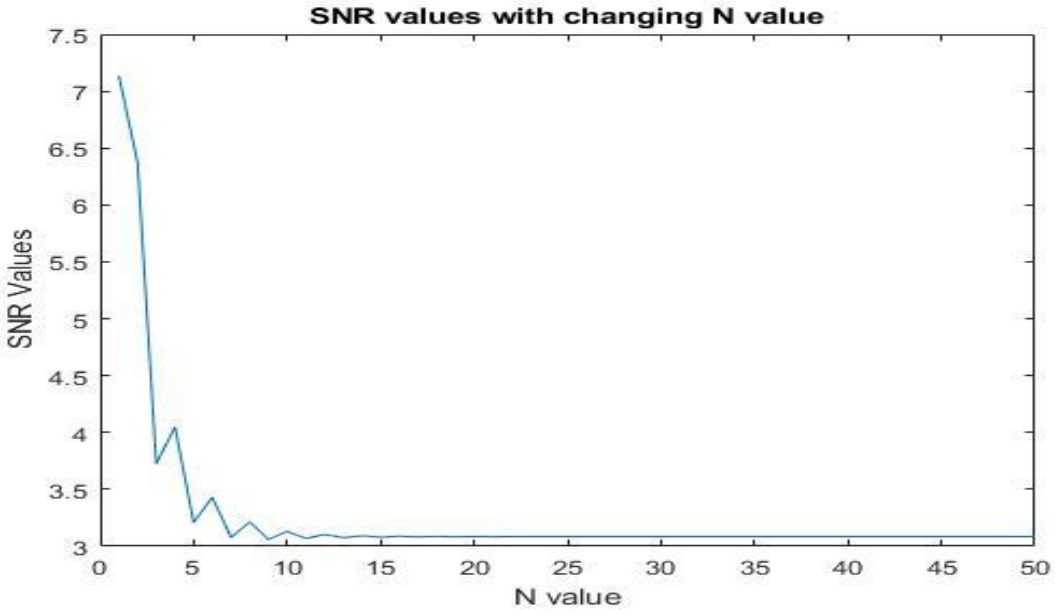
Problem 3:

In the problem 3, I just defined a function for implementing n tap filter. It may be inefficient or not working completely sufficient but it works well. I have calculated the SNR values varying with the parameters of the function.

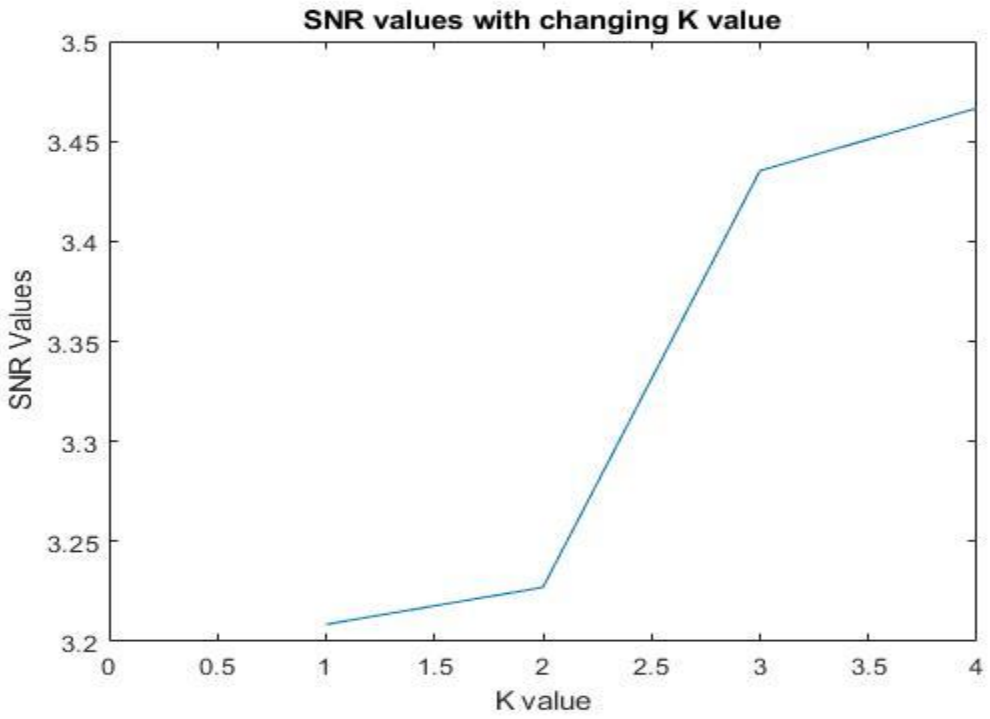
First figure is SNR value differentiating with a value the values on the x axis are *10^-2



---Second figure is SNR with respect to N values---



---Last figure is SNR with respect to K values(*10^2)---



There are merely 4 points in that figure (1,2,3,4) as in(100,200,300,400) ms.

```
File = 'mike.wav';
[x, Fs] = audioread(File);

SNRS = zeros(101,1);

%a values from 0 to 1 and plot SNR of mike.wav and recovered signal
for i=0:100

    drop = n_tap(i/100,5,100,x,Fs);
    Top=0;
    last=0;
    for j=1:length(x)
        Top = Top+ x(j)^2;
        last = last + (drop(j)-x(j))^2;
    end

    SNRS(i+1)= 10*log(Top/last);

end

plot(1:101,SNRS);
ylabel('SNR Values');
xlabel('*10^2 a value');
xlim([0 100]);
title('SNR values with changing a value');


figure;

SNRS =zeros(50,1);
```

```
%N from 1 to 50 and plot SNR of mike.wav and recovered signal
for i=1:50

    drop = n_tap(0.7,i,100,x,Fs);
    Top=0;
    last=0;
    for j=1:length(x)
        Top = Top+ x(j)^2;
        last = last + (drop(j)-x(j))^2;
    end

    SNRS(i)= 10*log(Top/last);

end

plot(1:50,SNRS);
ylabel('SNR Values');
xlabel('N value');
xlim([0 50]);
title('SNR values with changing N value');

figure;
SNRS =zeros(4,1);
%K between 100,200,300,400 miliseconds and plot SNR of mike.wav and recovered signal
for i=1:4

    drop = n_tap(0.7,5,100*i,x,Fs);
    Top=0;
    last=0;
    for j=1:length(x)
        Top = Top+ x(j)^2;
        last = last + (drop(j)-x(j))^2;
    end

    SNRS(i)= 10*log(Top/last);

end

plot(1:4,SNRS);
ylabel('SNR Values');
xlabel('K value');
xlim([0 4]);
title('SNR values with changing K value');

%Delay function

function y = n_tap(a,N,K,input,Freq)
Delay = Freq*K/1000;
L=length(input);
y=input;

for i=1:N
    for j=1:L
        if i*Delay+j <= L
            y(j+i*Delay)=y(j+i*Delay) + ((-a)^i)*input(j);
        end
    end
end


end
```