Hasan Utku Uçar
22102640
EEE321-2

Lab 1 Report

Part 1: Sound and plot signals corresponding to the note A in different octaves.

$$x_1(t) = \sin(2\pi f_o t).$$

Figure 1.a : Function corresponding to a note.

For different f_0 values, the pitch of the note changes accordingly. For example, when $f_0 = 440\ kHZ$ is put into the MatLab code provided, this signal yields the following plot.

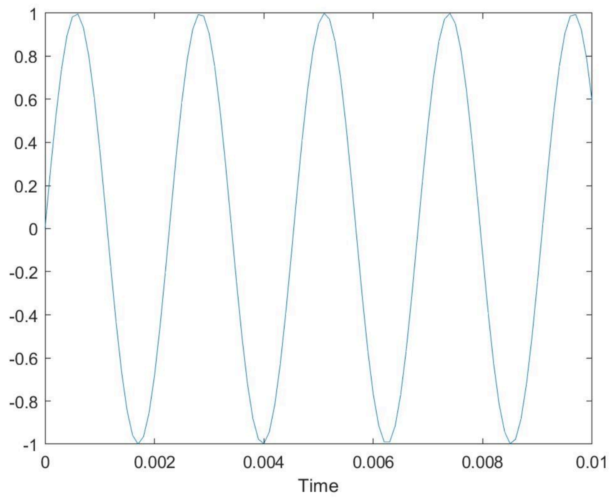
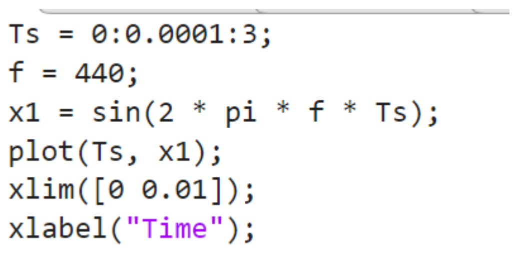


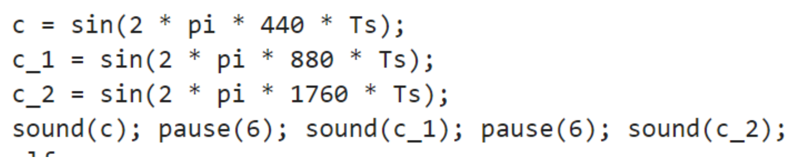
Figure 1.b : The output plot when $f_0 = 440\ kHZ$



```
Ts = 0:0.0001:3;
f = 440;
x1 = sin(2 * pi * f * Ts);
plot(Ts, x1);
xlim([0 0.01]);
xlabel("Time");
```

Figure 1.c : The Matlab code for this part

This experiment repeated for $2f_0$ and $4f_0$ causes the pitch to increase as the frequency increases.



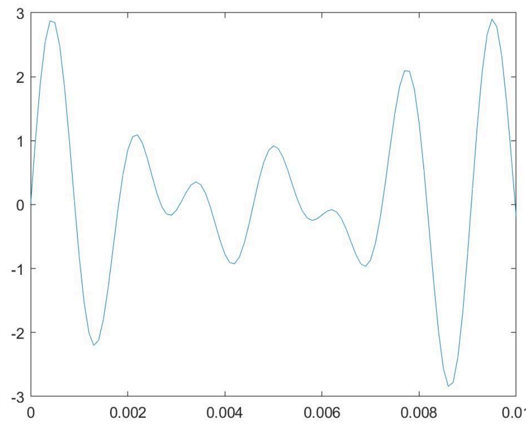
```
c = sin(2 * pi * 440 * Ts);
c_1 = sin(2 * pi * 880 * Ts);
c_2 = sin(2 * pi * 1760 * Ts);
sound(c); pause(6); sound(c_1); pause(6); sound(c_2);
```

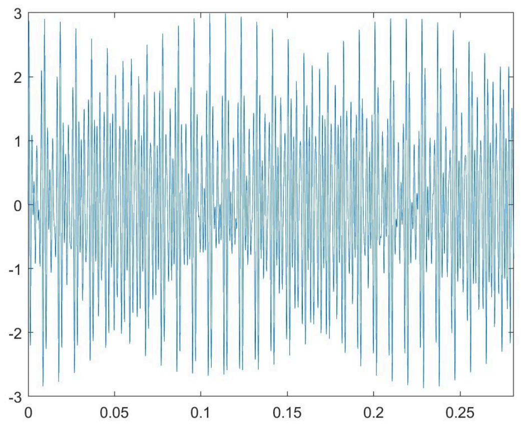
Figure 1.d : Matlab code to sound out the different octaves of A.

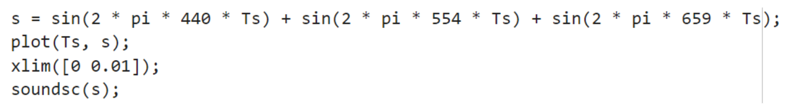
Combining the major triad (A, C#, and E) that can be expressed as:

$$s(t) = \sin(2\pi 440\, t) + \sin(2\pi 554\, t) + \sin(2\pi 659\, t)$$

The results are in the following plots using the code provided.





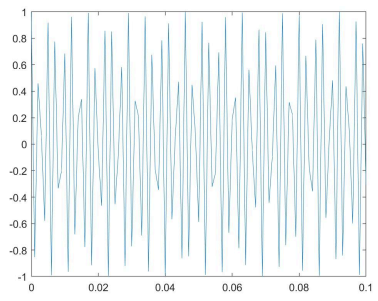


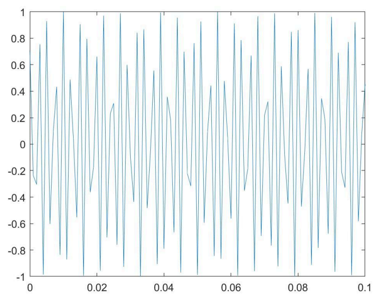
```
s = sin(2 * pi * 440 * Ts) + sin(2 * pi * 554 * Ts) + sin(2 * pi * 659 * Ts);
plot(Ts, s);
xlim([0 0.01]);
soundsc(s);
```

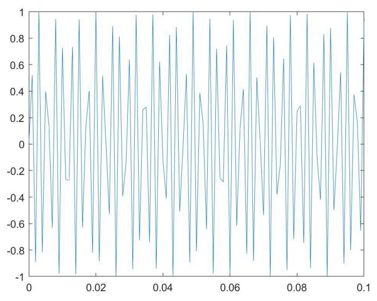
Part 2: The effect of Phase

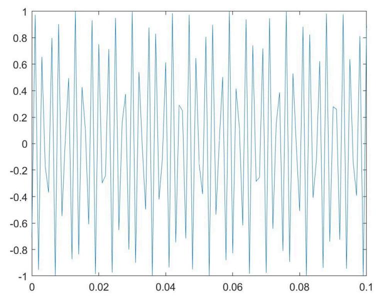
$$x_2(t) = \cos(2\pi f_o t + \phi) \text{ where } f_o = 587 \text{ Hz}$$

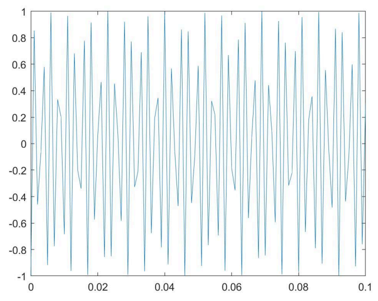
Incrementing phi up every $\pi/4$ radians results in the graph shifting but does not change the pitch or the volume of the sound, as phase only affects the time.



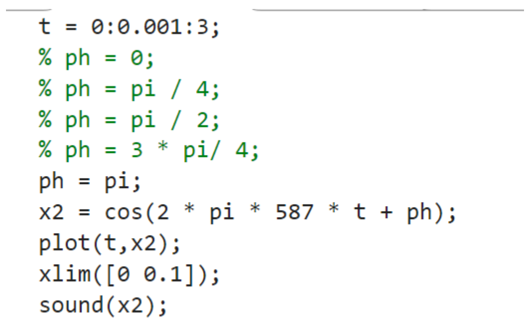








Figures 2.a-e: The effects of incrementing the ϕ parameter in the code from left to right phi is incremented at a step of $\pi/4$ radians starting from $\phi = 0$.



```
t = 0:0.001:3;
% ph = 0;
% ph = pi / 4;
% ph = pi / 2;
% ph = 3 * pi/ 4;
ph = pi;
x2 = cos(2 * pi * 587 * t + ph);
plot(t,x2);
xlim([0 0.1]);
sound(x2);
```

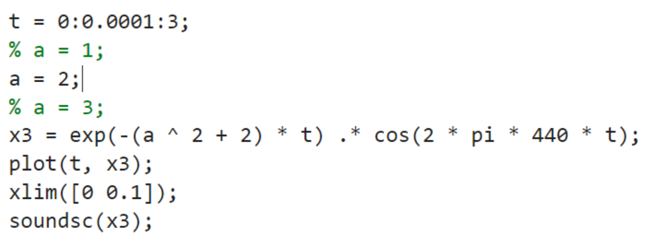
Figure 2.f : the code used for this part.

Part 3: Sinusoids with envelopes

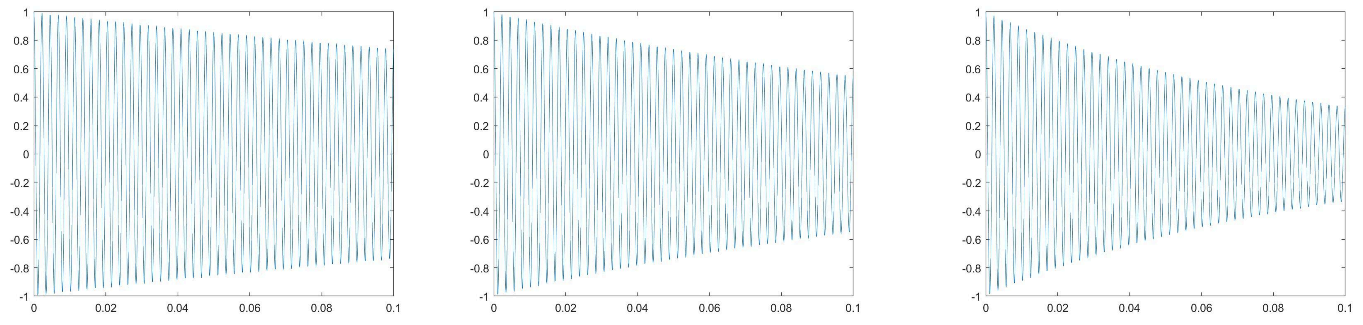
Considering the following sinusoids,

$$x_3(t) = e^{-(a^2+2)t} \cos(2\pi f_o t).$$

We want to see the effects of the change in the parameter “a.” Using the following Matlab code, we can see what happens when a is increased.



```
t = 0:0.0001:3;
% a = 1;
a = 2;
% a = 3;
x3 = exp(-(a ^ 2 + 2) * t) .* cos(2 * pi * 440 * t);
plot(t, x3);
xlim([0 0.1]);
soundsc(x3);
```



Figures 3.a-c: “a” is taken as 1, 2, 3, respectively, left to right.

We can observe that increasing the “a” value dampens the signal a lot faster.

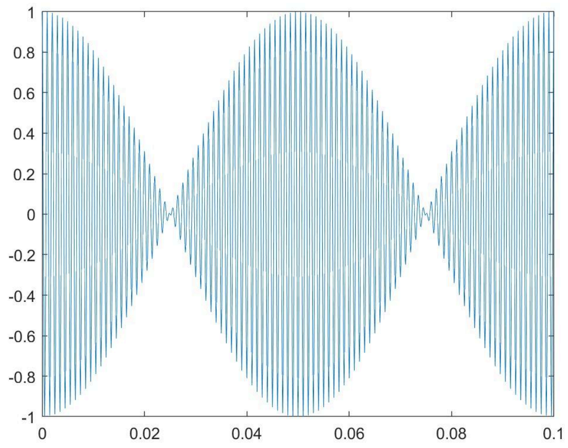
Part 4: Beat Notes and Amplitude Modulation

We want to observe the effects of multiplying sinusoids with different frequencies. For the first section of this part, the difference is very big, while in the second section, the difference is very small, and the trigonometric identity can be used in substitution for the multiplication. And lastly, the amplitude modulation that is used in radio frequencies is discussed.

$$x_4(t) = \cos(2\pi f_1 t)\cos(2\pi f_2 t)$$

$$\cos(\theta_1)\cos(\theta_2) = \frac{1}{2}\left[\cos(\theta_1+\theta_2)+\cos(\theta_1-\theta_2)\right],$$

$$x(t) = v(t)\cos(2\pi f_c t)$$



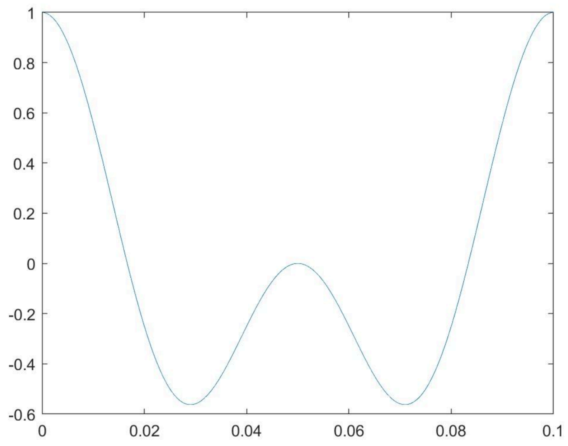
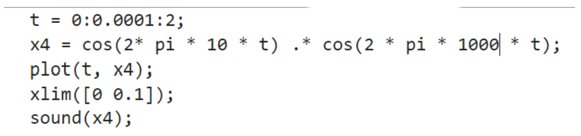


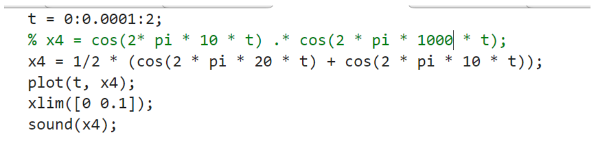
Figure 4.a and 4.b : When the frequency difference is large and small.

We can see that when the difference is very big, the smaller signal is carried along, and the bigger frequency acts as an envelope. Furthermore, when the difference is smaller, the output sound is like that of a “kick” from a drum -or a beat note-

Here is the code used for this part:



```
t = 0:0.0001:2;
x4 = cos(2* pi * 10 * t) .* cos(2 * pi * 1000 * t);
plot(t, x4);
xlim([0 0.1]);
sound(x4);
```



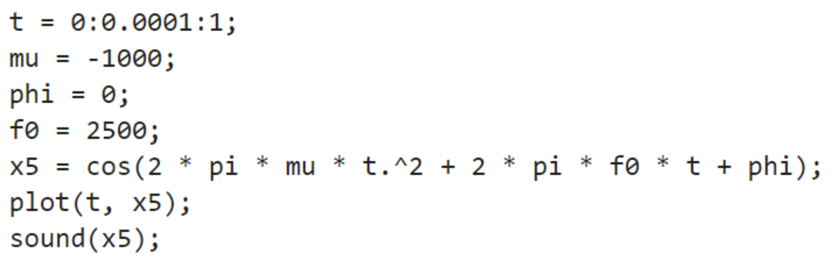
```
t = 0:0.0001:2;
% x4 = cos(2* pi * 10 * t) .* cos(2 * pi * 1000 * t);
x4 = 1/2 * (cos(2 * pi * 20 * t) + cos(2 * pi * 10 * t));
plot(t, x4);
xlim([0 0.1]);
sound(x4);
```

Part 5: Chirp Signals and Frequency Modulation

We want to observe the effects of time-varying frequency functions on the output itself.

$$x_5(t) = \cos(2\pi\mu t^2 + 2\pi f_o t + \phi)$$

Implementing this function in Matlab, where the μ value is calculated to be -1000 (Starting from 2500 to 500 over 2 seconds). The following code is used and yields the following plot.



```
t = 0:0.0001:1;
mu = -1000;
phi = 0;
f0 = 2500;
x5 = cos(2 * pi * mu * t.^2 + 2 * pi * f0 * t + phi);
plot(t, x5);
sound(x5);
```

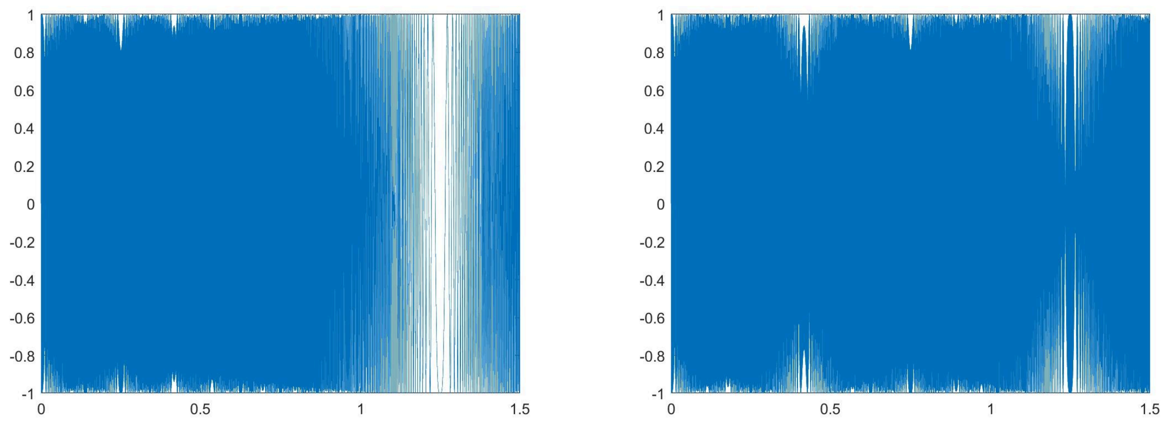
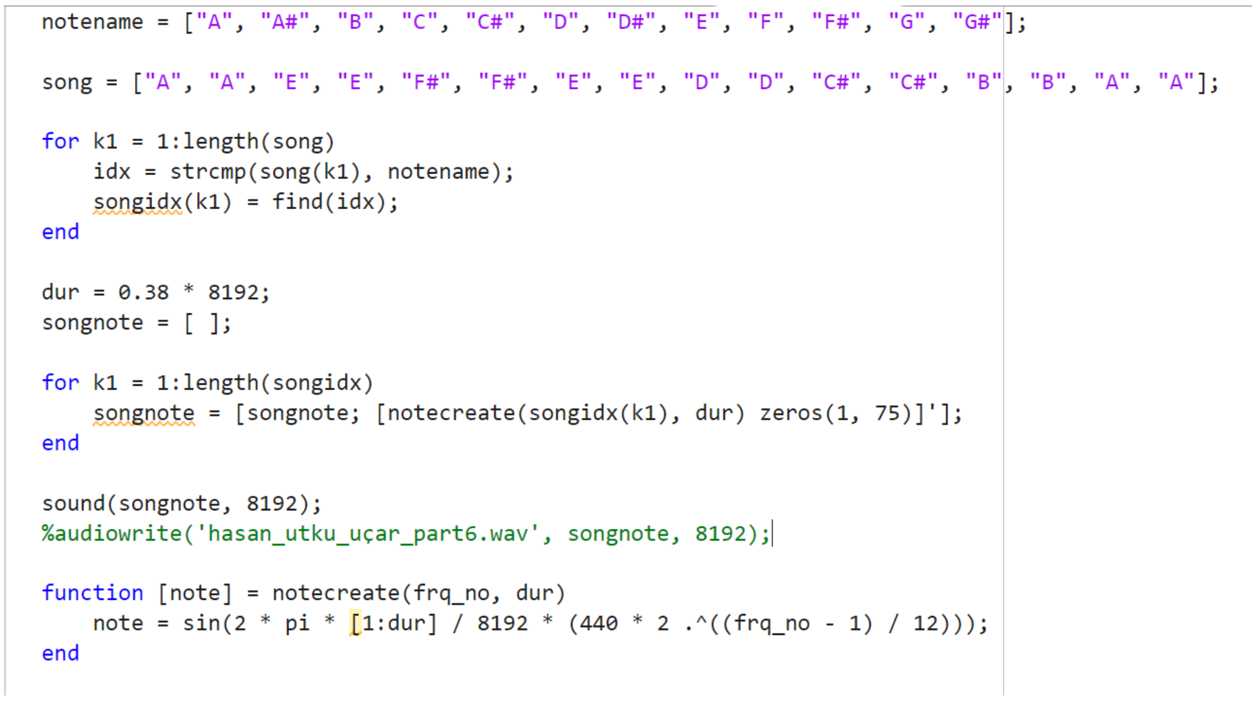


Figure 5.a-b: two different plots where the $\mu = -1000$ and $\mu = 1000$ respectively.

When the μ is -1000, while the volume stays the same, the pitch rapidly goes down, and the opposite happens when μ is 1000.

Part 6: Composing Music

In this part, we were tasked with first implementing a given code and then composing a song of our own. Here is the code we were tasked with implementing:

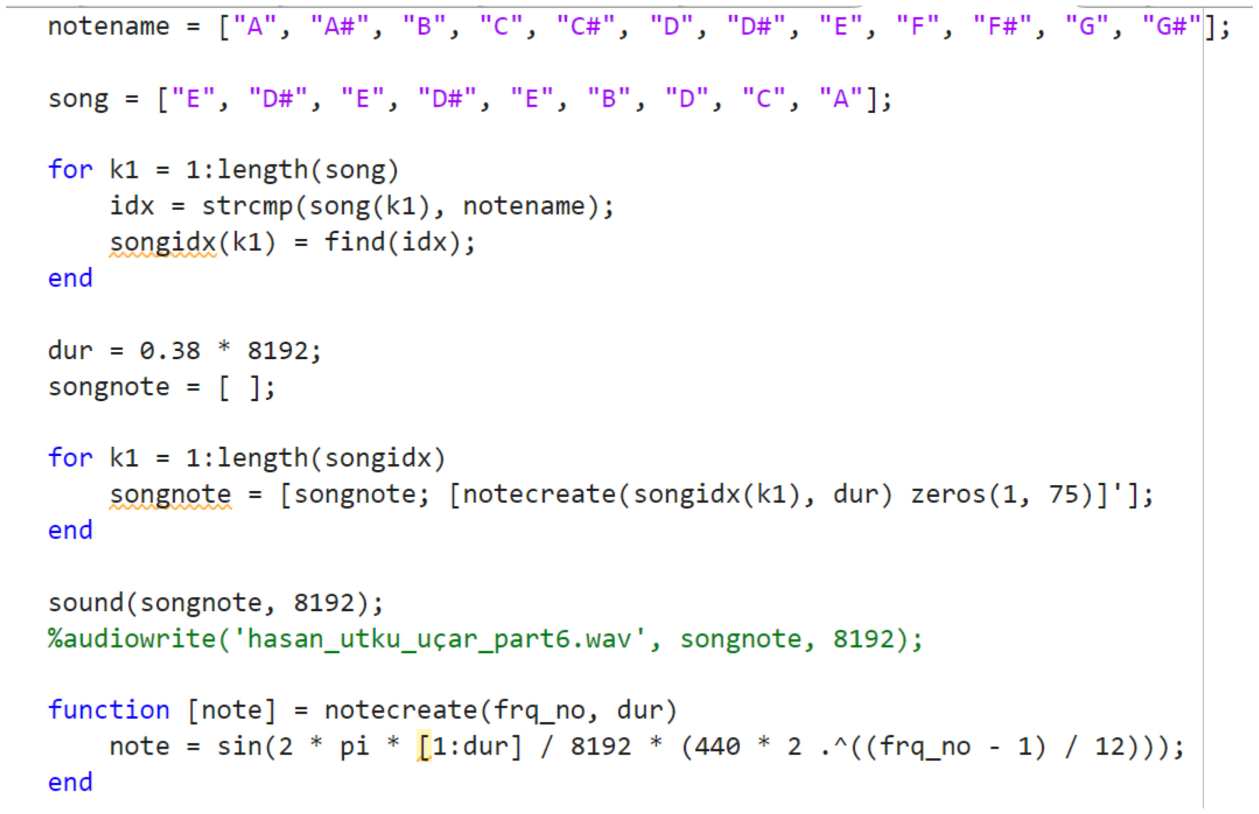


```
notename = ["A", "A#", "B", "C", "C#", "D", "D#", "E", "F", "F#", "G", "G#"];

song = ["A", "A", "E", "E", "F#", "F#", "E", "E", "D", "D", "C#", "C#", "B", "B", "A", "A"];

for k1 = 1:length(song)
    idx = strcmp(song(k1), notename);
    songidx(k1) = find(idx);
end

dur = 0.38 * 8192;
songnote = [ ];

for k1 = 1:length(songidx)
    songnote = [songnote; [notecreate(songidx(k1), dur) zeros(1, 75)]'];
end

sound(songnote, 8192);
%audiowrite('hasan_utku_uçar_part6.wav', songnote, 8192);

function [note] = notecreate(frq_no, dur)
    note = sin(2 * pi * [1:dur] / 8192 * (440 * 2 .^((frq_no - 1) / 12)));
end
```

Figure 6.a: The code for the first section of this part.

The function first looks up the corresponding data from the encrypted frequency numbers from the “notename” array. Then the song array creates a song using this encrypted data. Lastly “notecreate” function turns them into frequencies that can be heard. The output is the song we want to hear.

Here is the song I wanted to implement. It is the first 9 notes from Beethoven’s Für Elise. I wish to create the whole composition as soon as I learn to go down an octave between the notes.



```
notename = ["A", "A#", "B", "C", "C#", "D", "D#", "E", "F", "F#", "G", "G#"];

song = ["E", "D#", "E", "D#", "E", "B", "D", "C", "A"];

for k1 = 1:length(song)
    idx = strcmp(song(k1), notename);
    songidx(k1) = find(idx);
end

dur = 0.38 * 8192;
songnote = [ ];

for k1 = 1:length(songidx)
    songnote = [songnote; [notecreate(songidx(k1), dur) zeros(1, 75)]'];
end

sound(songnote, 8192);
%audiowrite('hasan_utku_uçar_part6.wav', songnote, 8192);

function [note] = notecreate(frq_no, dur)
    note = sin(2 * pi * [1:dur] / 8192 * (440 * 2 .^((frq_no - 1) / 12)));
end
```