```
diary on
format compact
%Johnny Li
%EEL3135 Fall 2018
%Lab 3 Part 1

%1.1
type key_to_note

function xx = key_to_note(X, keynum, dur)
%KEY_TO_NOTE Function created for lab3.1.
%Produce a desired note for a given duration at a given complex amplitude.

%{
    KEY_TO_NOTE: Produce a sinusoidal waveform corresponding to a given
    piano key number.
    Input Args:
        X: amplitude (default = 1)
        keynum: number of the note on piano keyboard
        dur: duration of the note (in seconds)
    Output:
        xx: sinusoidal waveform of the note
%}

%Code Given
%Smaple frequency
fs = 8000;
%Time interval
tt = 0:(1/fs):dur-1/fs;
%Given frequency function
freq = 440*2^((keynum-49)/12);

%Sinusoidal function
xx = real(X*exp(j*2*pi*freq*tt));
end

%1.2.1
type play_mary

%Plays a series of notes from mary.
%Script based on given instruction 1.2.

%Code given
% --------------play_mary.m-------------- %
mary.keys = [44 42 40 42 44 44 44 42 42 42 44 47 47];

%Notes: C D E F G
%Key #40 is middle-C

mary.durations = 0.25 * ones(1,length(mary.keys));
fs = 8000;  % 11025 Hz also works
xx= zeros(1, sum(mary.durations)*fs);

n1 = 1;
for kk = 1:length(mary.keys)
```

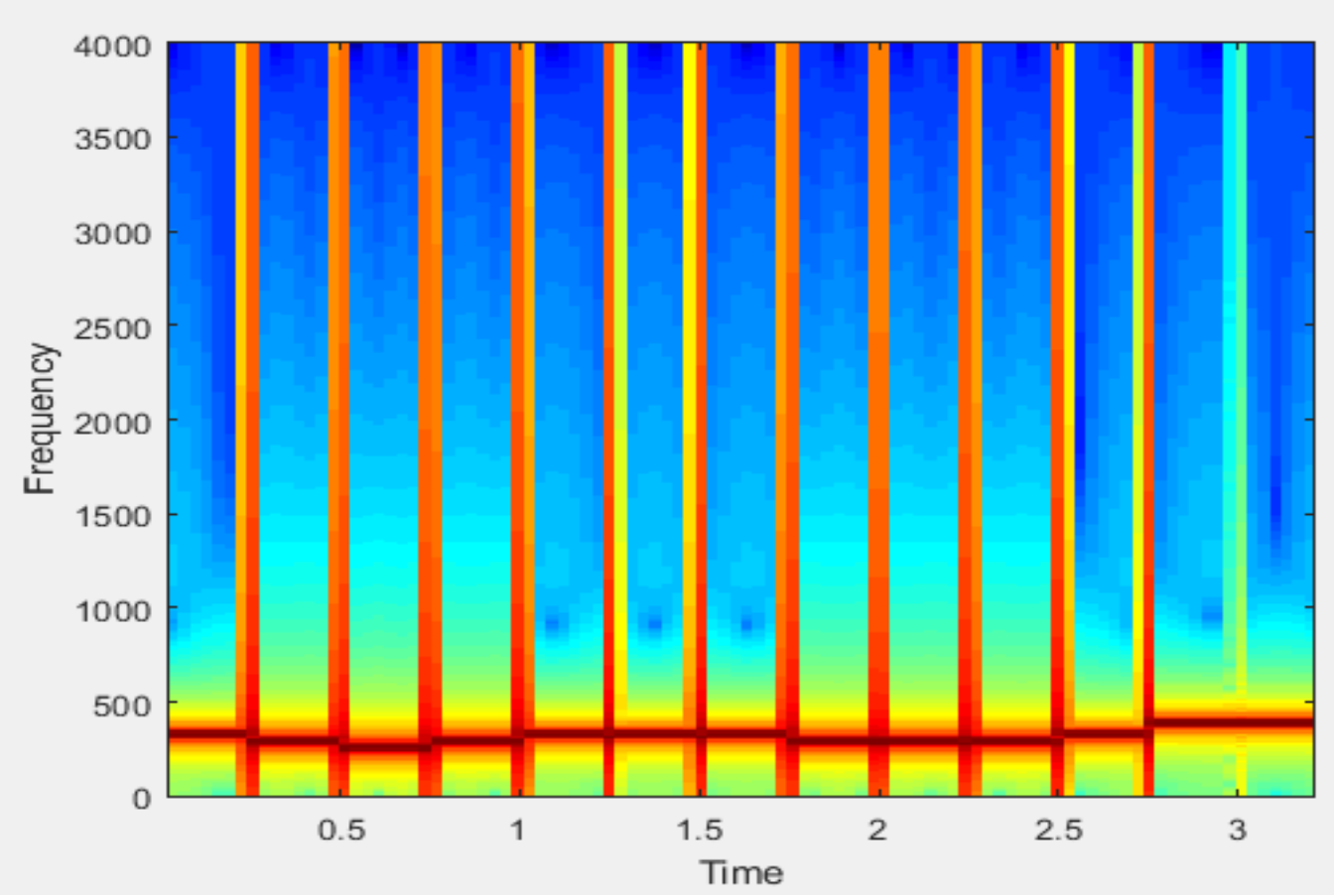
```
    keynum = mary.keys(kk);

    %Tone function
    tone = key_to_note(1,keynum,mary.durations(kk));

    n2 = n1 + length(tone) - 1;
    xx(n1:n2) = xx(n1:n2) + tone;
    n1 = n2 + 1;
end

%Create autofile
audiowrite('play_mary.wav',xx,fs);
play_mary

%1.2.2
%Plot the frequency-time spectrogram of Mary.
specgram(xx,512,fs)
```




```
%1.4
%Plays the first voice of the Barukh Fugue for 0.5 seconds each.
%Script based on given instruction 1.4.

%Code take from mary
% --------------play__first_voice_even.m-------------- %
```

```
%Notes: C D E F G
%Key #40 is middle-C

fs = 8000;  % 11025 Hz also works

%Edit functions
%Each note at 0.5 s each
dur=0.5*ones(1,length(theVoices(1).noteNumbers));
xx= zeros(1, sum(dur)*fs);

n1 = 1;
for kk = 1:length(theVoices(1).noteNumbers)
    keynum = theVoices(1).noteNumbers(kk);

    %Tone function
    tone = key_to_note(1,keynum,dur);

    n2 = n1 + length(tone) - 1;
    xx(n1:n2) = xx(n1:n2) + tone;
    n1 = n2 + 1;
end

%Create autofile
audiowrite('theVoices(1).wav',xx,fs);

play_first_voice_even

%1.5
type play_first_voice

% play each note in the first voice for itscorrect duration of pulses, with
each pulse being 0.15 seconds long.
%Script based on given instruction 1.5.

%Code take from play__first_voice_even
% --------------play__first_voice.m-------------- %

%Notes: C D E F G
%Key #40 is middle-C

fs = 8000;  % 11025 Hz also works

%Edit functions
%Each pulse at 0.15 s each
dur=0.15*theVoices(1).durations;
xx= zeros(1, sum(dur)*fs);

n1 = 1;
for kk = 1:length(theVoices(1).noteNumbers)
    keynum = theVoices(1).noteNumbers(kk);

    %Tone function
    tone = key_to_note(1,keynum,dur(kk));
```

```
    n2 = n1 + length(tone) - 1;
    xx(n1:n2) = xx(n1:n2) + tone;
    n1 = n2 + 1;
end

%Create autofile
audiowrite('playfirstvoice.wav',xx,fs);

play_first_voice


%1.6
type play_song

function song = playSong(theVoices)
%PLAY_SONG Construct the three voices in the Barukh Fugue and add them
%together. Add the five voices, from better_fugue.mat, together to produce
%the Better Fugue.
%Function based on given code 1.6.

%{
    PLAYSONG: Produce a sinusoidal waveform containing the combination of
    the different notes in theVoices
    Input Args:
        theVoices: structure contains noteNumbers, durations, and
        startpulses vectors for multiple voices of a song.
    Output:
        song: vector that represents discrete-time version of a musical
        waveform
    Usage:
        song = playSong()
%}
%load barukh_fugue.mat
load better_fugue.mat
%Define variables
%Frequency
fs = 8000;
%Beat per minute->beats per second->second per beats->second per pulse
%Given Code
beats_per_minute  = 120;
beats_per_second  = beats_per_minute / 60;
seconds_per_beat  = 1 / beats_per_second;
%spp = seconds_per_beat / 4;
%seconds per pulse, theVoices is measured in pulses with 4 pulses per beat

%Set spp to 0.15 for better fugue
spp=0.15;

%Length of voices
numV=length(theVoices);
%Length of notes
numN=length(theVoices(numV).noteNumbers);
%Final start pulse
fsp=theVoices(numV).startPulses(numN);
```

```
%Final durations
fd=theVoices(numV).durations(numN);
song = zeros(1,ceil((fsp+fd)*spp*fs));

%Get Max value in theVoices
M=0;
for a=1:length(theVoices)
    for b=1:length(theVoices(a).durations)
        d=theVoices(a).durations(a);
        st=theVoices(a).startPulses(b);
        if M<(d+st)
            M=d+st+1;
        end
    end
end

%Longest value in better
song = zeros(1,ceil(M*spp*fs));
%Create a vector of zeros with length equal to the total number of samples
%in the entire song

%Then add in the notes
for i = 1:length(theVoices)
    for j = 1:length(theVoices(i).noteNumbers)
         note =
key_to_note(1,theVoices(i).noteNumbers(j),theVoices(i).durations(j)*spp);
         %Create sinusoid of correct length to represent a single note
         locstart = theVoices(i).startPulses(j)*spp*fs;
         %Index of where note starts
         locend = locstart+length(note)-1;
         %Index of where note ends
         song(locstart:locend) = song(locstart:locend) + note;
    end
%For clipping
song=song/(max(abs(song)));
end
%Create autofile
audiowrite('better_fugue1.wav',song,fs);
end

%With barukh fugue
play_song(theVoices);
%With better fugue
play_song(theVoices);

diary off
```