```
%% LABORATORY PRACTICUM #0
% Course: EEL3135 (Written By: Professor Harley)

clear;

%% SCRIPT
% This script takes a name (my_name variable) and translates it into
% music. Specifically, each character in the name turns into a single note.
% In the script, users can easily manipulate the frequency multiplier
% (f_multiplier) for each note, the duration (my_duration) for each note,
% and the sampling rate of the music (fs).

% ENTER YOUR NAME IN THE NEXT LINE
my_name = 'Natalie';
N = length(my_name);        % Number of characters in name

% DEFINE CONSTANT
f_multiplier = 2;               % Frequency multiplier (unitless)
my_duration  = 0.5;             % Note duration (in seconds)
fs = 44100;                     % Sampling rate (in samples / s)

% DEFINE NOTE FREUQENCY AND DURATION
cnote = double(my_name)*2;      % Note frequencies (in Hz)
dur = my_duration*fs*ones(N,1); % Note durations (in samples)
disp(['The Duration is: ' num2str(dur.')])

% CREATE MUSIC
z = cell2mat(arrayfun(@(n) simple_sawtooth(cnote(n), dur(n), fs), 1:N, 'UniformOutput',↙
false )).';

% PLOT OUTPUT WAVEFORM
t = 1/fs:1/fs:N*dur(1)/fs;      % Time Axis
figure(1)
plot(t, z)                      % Plot
xlabel('Time [s]')              % Horizontal axis label
ylabel('Music Amplitude')       % Vertical axis label
title('Full Music Signal')      % Plot title

figure(2)
plot(t, z)                 % Plot
xlim([0 dur(1)/fs])
xlabel('Time [s]')         % Horizontal axis label
ylabel('Music Amplitude')  % Vertical axis label
title('One Note of Music Signal')    % Plot title

% PLAY MUSIC
soundsc(z, fs)             % Play music

% SAVE MUSIC
audiowrite([my_name '.wav'], z./max(z)*0.99, fs)
```

```
%% FUNCTIONS USED IN THE CODE
function z = simple_sawtooth(note, dur, fs)
%SIMPLE_SAWTOOTH  Create a single sawtooth wave note
%   Z = SIMPLE_SAWTOOTH(NOTE,DUR,FS) create a time-modulated sawtooth
%   note, where NOTE is the note frequency (in Hz), DUR is the note
%   duration (in samples), and FS is the sampling frequency.
%
%   see also: sawtooth
%

    % BUILD INITIAL SAWTOOTH
    z = sawtooth(2*pi*note*(0:dur-1)/fs,0.2);
    Pattack  = .3;             % Length of attack  (proportion)
    Prelease = 1-Pattack;      % Length of release (proportion)

    Llength  = numel(z);       % Length of signal

    % SET LOW FREQEUNCY SIGNAL TO MODULATE WITH SAWTOOTH
    Lattack  = floor(Llength*Pattack);                      % Length of attack
    Lrelease = ceil(Llength*Prelease);                      % Legnth of release

    Vattack  = 1;                                           % Attack maximum value
    Vrelease = 0.5;                                         % Sustain value

    attack  = linspace(0, Vattack, Lattack);                % Attack time weights
    release = linspace(Vrelease, 0, Lrelease);              % Release time weights

    weight = [attack,release];                              % Concatenate
    weight = conv(weight,exp(-0.00001*(1:1000)), 'same');    % Smooth everything

    % APPLY MODULATION
    z = z.*weight;                                          % Output signal

end

function y = sawtooth(t,width)

    t0 = t / (2*pi);
    y = 2*(t0-floor(t0))-1;

end
```