```
% returns the raw audio of the nth C major note
% for the given durration of time
% this function could be modified to take the scale as an argument and the
% for loop below could extract the necessary frequencies for that scale.
% INPUT:
% n: the index of the note to play
% durr: the duration of the note in seconds
function [s,notesCMaj] = note(n,durr)
    sampleRate = 44*1024;
    numNotesOnPiano = 88;

    % extracting just the "white keys" on the piano
    notesCMaj = [];
    j = 1;
    for i=4:12:numNotesOnPiano-12
        notesCMaj(j)   = notes(i);
        notesCMaj(j+1) = notes(i+2);
        notesCMaj(j+2) = notes(i+4);
        notesCMaj(j+3) = notes(i+5);
        notesCMaj(j+4) = notes(i+7);
        notesCMaj(j+5) = notes(i+9);
        notesCMaj(j+6) = notes(i+11);
        j = j + 7;
    end

   % possible special case for silence (but this doesn't work with music
   % interpolation because notes whose index is near 0 bear not special
   % relation to silence because the lowest note on the piano is 27.5 hz,
   % and silence is 0 Hz.
   if n == 0
       f = 0;
   else
       f = notesCMaj(n);
   end

   % generating the piano note as a sound wave with the given frequncy
   s = snd(f,sampleRate,durr);

    % fade the sound out
    % this is the same as multiplying by a diagonal matrix whose diagonal is a
    % discrete linear function
    for i=1:length(s)
        s(i) = s(i)*(length(s)-i-1)/length(s);
    end

```