Name: SUN RUI
Student ID: 18083229g

Note: the programs, all source audio files and creating audio files are in the same folder.

For three questions, my answers:

1.1 sound function: the audio is played in the background directly;
audioplayer function: we can use it to control the the playing process, such as play, pause, resume, stop and so on.

1.2 WAVE (.wav), OGG (.ogg), FLAC (.flac), MPEG-4 AAC（.m4a、.mp4）

1.3 22050HZ

For task1, which is to show the default sample rate of clap.mp3: (task1_sample_rate.m)

In this task, we use function audioread to read the source file, from this we can get variable y of content and variable f of sample rate. Finally, we use function fprintf to print the sample rate to terminal, which is "sample rate:22050Hz", the code:

```
filename = "clap.mp3";
[y,f] = audioread(filename);    % read clap.mp3 file
fprintf("sample rate:%dHz\n",f);
```

For task2, which is to speed clap.mp3 by 2x: (task2_speed2x.m)

Step 1, we need to use the function "audioread" to open the source file, which can return content and frequency of the source audio file, the code:

```
filename = "clap.mp3";
[y,f] = audioread(filename);    % read clap.mp3 file
```

Step 2, when we use function "audiowrite" to write new audio file, the parameter frequency value should be double. Because the frequency is twice than before, the speed also would be twice than before, the code:

```
filename = "clap2.flac";
audiowrite(filename, y, f*2);    % frequency * 2 to speed
```

For task3, which is to show clap.mp3's frequency spectrum: (task3_spectrum.m)

Step 1, we need to use the function "audioread" to open the source file, which can return content and frequency of the source audio file, the code:

```
filename = "clap.mp3";
[y,f] = audioread(filename);    % read clap.mp3 file
```

Step 2, we use function fft(y) which is the discrete Fourier transform of the vector y, through some improvements, we get x axis of frequency and y axis of intension, then use function plot to draw the picture, the code:

```
%%%%% draw frequency spectrum pic %%%%%
Y = fft(y);
L = length(y);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = f*(0:(L/2))/L;
plot(f,P1)
```

Step 3, we need to set the name of x axis, name of y axis and a title of the spectrum picture. The spectrum picture is task3_spectrum.jpg in my zip package, the code:

```
title('clap.mp3 ¡s frequency spectrum')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

For task4, which is to merge three different audio files into a single audio file and fade it in and out: (task4_merge_and_fade.m)

Step 1, we need to open all three files by function audioread and get their content and frequencies, here omit codes.

Step 2, we need to search the highest frequency among these three audio files and resample them with this frequency. In this step we get a new frequency and three content values which are important in next steps. The code:

```
f = max(f1, f2);
f = max(f, f3);    % find the max frequency

% resample them
y1 = resample(y1, f1, f);
y2 = resample(y2, f2, f);
y3 = resample(y3, f3, f);
```

Step 3, we start to merge three audio files, the method just is to put their content values into a list in order, the code:

```
% merge them
y = [y1; y2; y3];
```

Step 4, we add 3s fade-in and 3s fade-out to the merging audio file. In the 3s head of the audio, the sound may become bigger gradually, and the sound may become smaller gradually at the 3s end. In order to realize this effect, we use an increasing float array and a decreasing float array to multiply 3s-front part and 3s-ending part of the new content value, the code:

```
% fade in and out
in_length = 3*f; % 3s fade in
out_length = 3*f; % 3s fade out

% prepare masks
in_mask = (1:in_length)' / in_length;    % small to big
out_mask = (out_length:-1:1)' / out_length;    % big to small

% apply mask
y(1:in_length) = y(1:in_length) .* in_mask;    % change 3s head
y(end - out_length + 1:end) = y(end - out_length + 1:end) .* out_mask;    %
change 3s end
```

Step 5, we use function audiowrite to save the new audio file, here omit codes.