Prelab: Before starting your m-file, read through this entire document, then create a pseudocode flowchart of what your program will do using the standard symbols given in Section 8.2 of your textbook.

Deliverables: The following four files need to be uploaded to D2L once the lab is complete.

- **lab5.m** your code from Lab 5 divided into logical sections and appropriatly commented with description
- **bands13.wav**: a sound file that contains bands 1 and 3
- **band2.wav**: a sound file contains band 2 only
- **pcode.pdf**: a PDF scan of your pseudocode for this assignment

Lab: Create a new m-file called **lab5.m**. Begin the file with a header section starting with a %% line that contains the lab title and your name. Then % lines which show Name, Course, Lab, and Date. Then start a new section with %% and start the code with the usual **clear, clc, close all** commands, followed by the loading of the **filter_coeffs.mat** file provided by your instructor. This file provides 3 matrices: **lpf**, **bpf**, and **hpf**. Copy the **rdd.wav** file (also provided by your instructor) to your Matlab folder. You can load the wav file into memory with the following command:

```
[z, fs] = audioread('rdd.wav')
```

The result should be a 2-column matrix **z** that contains the sound samples versus time. It contains 2 channels. The left channel is column 1 and right channel is column2. This is a 16-bit, signed (as in +/-) wav file with a sampling rate of 44.1kHz. **fs** should contain the sampling rate.

You can listen to the file within Matlab with the following command:

```
sound(z, fs)
```

To stop playback, use the **clear all** command.

Your **lab05.m** file will run with a user at the command prompt. The user will not be able to alter your program's code. They will only be able to run it. The code will perform the following tasks:

- Select either the left or right channel, depending on user input.
- Select frequency band(s) chosen by the user to keep. The band(s) not selected will be filtered out of the final output.
- Combine each of the selected bands together. Each band will be a vector. Multiple bands can be combined by adding the vectors.
- If the user chooses, write the resulting, filtered results to a .wav file using Matlab's **audiowrite** command. You may need to use the **wavwrite** command instead if you have an older version of Matlab. Make sure the resulting code runs on the version of Matlab used in class (R2015a). The user will choose what the file name will be.

In audio processing, filtering is picking out the frequencies you want to keep and leaving the rest behind. For example, low-pass filtering allows low frequencies to pass through (the ones we keep) and mostly gets rid of the higher frequencies. 3 filters will be used to produce 3 different bands:

- **band1**: will contain the low frequencies (lower than 300Hz) and will be the result of applying the low-pass filter described by **lpf**.
- **band2**: will contain the mid-range frequencies (from 300Hz to 3kHz) and will be the result of applying the bandpass filter described by **bpf**.
- **band3**: will contain the high frequencies (above 3kHz) and will be the result of applying the high-pass filter described by **hpf**.

For example, assuming you've already selected either the right or left channel and have stored that result in the column vector **ch**, applying the low-pass filter will look something like the following:

band1 = filter(second row of **lpf**, first row of **lpf**, ch);

Of course, you will have to convert parts of this statement into actual Matlab code. The result of applying the low-pass filter is stored in the **band1** variable. If the user selects more than one band, the bands can be combined by simply summing them, as each band will be column vectors of the same length.

Store your filtered results in a variable called **filtered**. You will have to research how to write your **filtered** vector to a wav file.

Below is a sample of how your user interface should look. You should make your user interface look exactly like this, down to the quote marks. The things inputted by the user are in **bold print**. Your script should run with the user input exactly as shown. For instance, they will use brackets when specifying multiple bands, and they will not put quotes around their yes/no response.

Choose which channel you would like to filter (1 for left, 2 for right): **2**
Right channel selected.

Choose which of 3 bands you would like in your filtered wav file: **[1 3]**
Including band 1.
Including band 3.

Would you like to save your filtered results to a file (y/n)? **y**
Enter the file name (do not include the .wav extension): **bands13**
Successfully wrote 'bands13.wav' to a file.

>>

Listen to each band and hear the difference between the different frequency bands. Each frequency band will sound very different than the others. After making sure your user interface and output is right, submit the following files via D2L:

- **lab5.m** Your code from Lab5. Ensure it is logically divided into sections usuing %% and has % commented descriptions that explain what the program does at each step.
- **bands13.wav**: contains bands 1 and 3
- **band2.wav**: contains band 2 only
- **pcode.pdf**: a PDF scan of your pseudocode for this assignment