ECE 3331, Dr. Hebert, Fall 2023 Programming Assignment 10 due Saturday 11/04 at 11:59 pm

In this project, you will write a program that can take any stereo 16-bit WAV file with sample rate 44,100 samples per second as input and write out a stereo 16-bit WAV file that plays the stereo sound backwards, and at half-speed.
The WAVE header of the output file will contain two values that are different from the values in the header of the input WAVE file: Samplerate and Byterate.
The Samplerate and Byterate in the output file will be one-half the corresponding values in the input file. The input file will be a stereo WAVE file, so the left-channel and right-channel audio samples are interleaved: one left-channel sample, then one right-channel sample, then one left-channel sample, ….etc.
When you fwrite( ) the audiosamples in the input WAVE file to the output WAVE file in reverse order, the output file must also have the same interleaving of left-channel and right-channel samples.

A) print out instructions to the user, asking the user for the name of an input stereo 16-bit WAV file, and the name for an output stereo 16-bit WAV file.
B) Open the input file as "rb". If the input filename entered by the user fails to open correctly (fp == NULL) or the wav file has the wrong samplerate (44,100 required) , or is mono instead of stereo, inform the user and ask for another file. Repeat until the requirements are met.
C) Open the output file as "w+b". Since the input file wav header and output file wav header will be identical except for the two fields, copy the wav header from the input file into the output file (fread() then fwrite()). Then fseek( ) in the output file to the Samplerate and Byterate fields and decrease their values in the output file by ½.
D) Run-time allocate (malloc( ) or calloc( )) a 2-D array data[ ][ ] to hold the number of bytes of16-bit audio data.
The array should have 2 **rows**, and enough **columns** to hold all of the samples for the left/right channels in the stereo audio data.
Note: stereo data is stored as 1 left channel sample, then 1 right-channel sample, then 1 left-channel sample, then 1 right-channel sample, etc… etc. (See chapt 11 for run-time allocation of 2-D arrays). For example, samplerate= 44100, bitspersample=16, and subchunk2size= 8800; then the number of audiosamples (a left-channel sample plus a right channel sample = 1 stereo audio data sample) is 8800/(2*2)=2200 audio samples. The total number of bytes inn the data subchunk = 8800, and each left or right sample = 2 and there are 2 channels (left and right).
E) Read in the stereo data into your 2-D array. But be careful.
Note that when you call the ansi C function malloc( ) , it returns a sequential block of memory.
The next time you call malloc( ), you also get a sequential block of memory. However, this 2nd block of memory might not be connected to the 1st block.
Think about this in regards to malloc( )'ing the 1-D row vectors of a 2-D array.
The block of memory for the 1st row might not be connected to the block of memory for the 0th row.
This is different from when you declare a fixed size 2-D array.
When declaring a fixed size 2-D array float arr2[2][1000], the cells are stored in a single connected block of memory.
But when doing dynamic allocation of float arr2[ ][ ] using

```
float ** arr2;
 arr2 =(float **)malloc( 2*sizeof(float *) );
 for(i=0; i<2; i++){
   arr2[i] = (float*)malloc(1000*sizeof(float));
 }
```

the blocks of memory for each of the two rows need not be connected.
Therefore, to read data into this float array, you can't simply do a single
fread(&arr2[0][0],4,2*1000,fp); into a dynamically allocated 2-D array, because the start in memory for the 1st row might have a gap from the end of the 0th row in memory.
It might or might-not work, depending on where the operating system "chose" to store each row of the 2-D dynamically allocated array.
If the fread( ) fails because the rows are not in connected blocks, your program will not be aware of the failure, unless you capture the return value from fread( ).

```
int iret;
iret=fread(&arr2[i][0],4,1,fp);
```

Here iret captures the number of cells that fread( ) succesfully read into the ith row of 2-D array arr2[ ] [ ].
So, in your program, read-in the wav data from the file into your 2-D short int array row-by-row using a for( ) loop, instead of a single big fread( ) to read-in the complete set of audio data.
F) Now write the audio data into the output file in reverse order:
data[N-1][0], data[N-1][1], data[N-2][0], data[N-2][1], …..,data[0][0], data[0][1].

```
for(i= numberaudiosamples-1; i>=0; i--){
  for(j=0; j<2; j++){
 // fwrite( ) one left channel value
 //  fwrite( ) one right channel value
  }
}
```

G) fclose( ) the wav files.

H) Free the run-time allocated memory.

You are done!