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

You will write a program to generate a WAV mono file that plays a short song as discussed below.
You will name the output wav file "song.wav" and you will store it in the same directory from which your code is executed.

1) Define a struct that will hold a wav file header.

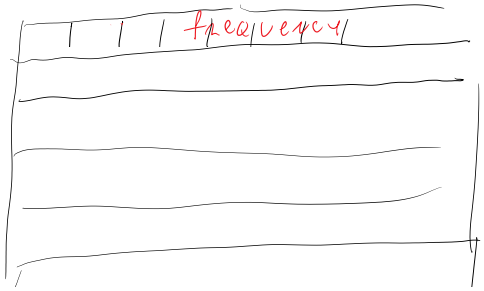
```
struct header{
  char chunkid[4];
  unsigned int chunksize;
  char chunkformat[4];
   etc, etc,.....
};
```

2) Declare a variable of type struct header, e.g.
struct header HeaderOut.
3) Fill-in the fields of HeaderOut.
For example, HeaderOut.chunkid[ ] should hold 'R' 'I' 'F' 'F', etc
Use:
sample rate: 44100
number channels: 1
bitspersample: 16
Some of the fields (e.g. chunksize) will not be known until you write the audio data into the file.
But fill-in all other fields as possible.
4) Open up the file "song.wav" as "wb" in the same directory from which your code is executed.
Check if the file was successfully opened; if not notify the user, then exit the program.
5) Write the wav header into the file,even though you have not filled in all fields. Later you will count the bytes in the audio data as you fwrite( ) it into the file. Then you will fseek( ) back to the header and complete the entries.
6) Use malloc( ) as shown in class and in your textbook, to create a 2D array to hold signed short int audio samples .
Each row of the array will hold 1/3 of a second of a sine wave.
Make the 2D array to be of data type short int with 25 rows and 44100*(1/3) columns.
Recall, first declare a pointer-to-pointer-to-short int for the name of the 2D array.
Then, malloc( ) a 1-D array of pointer-to-short int to hold pointers to each of the rows.
Then using a for loop, malloc ( ) memory for each row of the 2D array.
7) Load the following notes (sine wave) into the first 24 rows of the 2D array , using an amplitude of 32,700.0 for the sin waves.
The sampling rate is 44,100 samples per second. So 32700.0*sin(2*pi*freq*t) should have time-samples at t=0.0; t=1.0/44100.0, t=2.0/44100.0, etc.

Row	Note	Freq (Hz)
0	A 4	440.00
1	$A^{\#}$ 4	466.16
2	B 4	493.88
3	C 4	523.25
4	$C^{\#}$ 4	554.37
5	D 4	587.33
6	$D^{\#}$ 4	622.25
7	E 4	659.25
8	F 4	698.46
9	$F^{\#}$ 4	739.99
10	G 4	783.99
11	$G^{\#}$ 4	830.61
12	A 5	880.0
13	$A^{\#}$ 5	932.33
14	B 5	987.77
15	C 5	1046.50
16	$C^{\#}$ 5	1109.73
17	D 5	1174.66
18	$D^{\#}$ 5	1244.51
19	E 5	1318.51
20	F 5	1396.91
21	$F^{\#}$ 5	1479.98
22	G 5	1567.98
23	$G^{\#}$ 5	1661.22

24	silence	all 0's

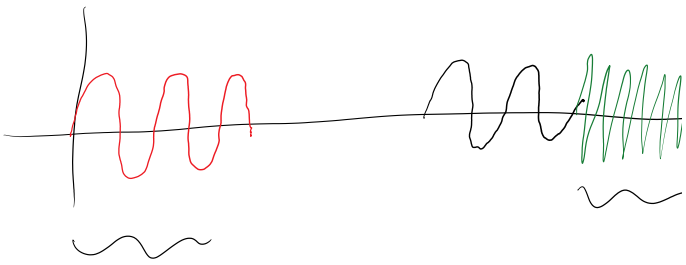
Your 2-D array will look something like this :



8) Now a song can be defined as a sequence of row numbers (row numbers 0 through 24).
For example, the 48 values
3,3,10,10,12,12,10,24
8,8,7,7,5,5,3,24
10,10,8,8,7,7,5,24
10,10,8,8,7,7,5,24
3,3,10,10,12,12,10,24
8,8,7,7,5,5,3,3
should play the children's song "Baa Baa Black Sheep"

You can use
unsigned short int song[48]=
{3,3,10,10,12,12,10,24,8,8,7,7,5,5,3,24,10,10,8,8,7,7,5,24,10,10,8,8,7,7,5,24,3,3,10,10,12,12,10,24,8,8,7,7,5,5,3,24};
to define the song.
9) Loop through the row numbers contained in the elements of song [ ].
For song[0]=3 you will write row 3 of your 2D array into the wav file for the 0th note.
For song[1]=3 you will write row 3 of your 2D array into the wav file for the 1st note.
For song[2]=10 you will write row 10 of your 2D array into the wav file for the 2nd note.
etc…
Keep a count of the total number of bytes written into the file.

The data subchunk will look something like this :



10) Now fseek( ) in the file "song.wav" back to the header fields that were not filled-in. Compute their correct values and fill-in those fields.
11) fclose( ) the file and free( ) the malloc( )'d memory.

Your project is correct when double clicking on your file results in the song "Baa Baa Black Sheep" being played.