Problem Set 4
Total points: 80

Note that the following URL provides a good reference for C language library function usage:
http://www.cplusplus.com/reference/cstdlib/

1. Command Line Parsing (20 points)

Create a program that has the following command line usage:

```
main –max level_dBFS  | -rms level_dBFS ifile ofile
where
level_dBFS is the maximum absolute level in ofile in dB relative
to Full Scale expressed as a non-positive integer
rms_dBFS is the Root Mean Squre (rms) level in ofile in dB
relative to Full Scale, expressed as a non-positive integer
ifile is the input audio file in *.wav format
ofile is the output audio file in *.wav format
```

If the correct number of arguments are not present, the program prints the usage lines shown above. The "|" above indicates that either –max or –rms and the associated arg must be present, but not both

Consider using strncmp() to determine if –max or –rms are present on command line. This requires #include <string.h>. Consider using atoi() to convert an arg string to a integer value. This requires #include <stdlib.h>

2. Binary file I/O (20 points)
Use the program in problem (1). Assume that ifile and ofile are in 16-bit PCM WAV format, single channel. Use the supplied file `tone1k.wav` as ifile.
Assume the simplest structure of a WAV file: that it has a 44-byte header followed by a singe chunk of audio signal data. More information about the WAV header is at
http://soundfile.sapp.org/doc/WaveFormat/
You might also want to review Big-Endian and Little-Endian at
https://en.wikipedia.org/wiki/Endianness

Open ifile for binary reading:

```
if ( (ifp = fopen(ifile, "r")) == NULL) { … error reporting
```

using error checking and reporting.

Read 44 bytes into an unsigned character array using error checking and reporting. This is the WAV header.

```
unsigned char header[44];
if ( fread(header, sizeof(header), 1, ifp) != 1 ) { … error
reporting
```

Print each byte of the wav header as character and hexadecimal values as 4 lies of 11 characters per line using this format per character:

```
printf("%c %02x ", c, header[i]);
```
where if header[i] is an ASCII character, then print the character, otherwise print a SPACE character. This could be done using:
```
c = isalpha(header[i-1]) ? header[i-1] : ' ';
```
which requires #include <ctype.h>.

Confirm that this is a valid WAV header by visual inspection.

Print:

- Number of channels
- Sampling Rate
- Bits per sample
- The data size, in bytes, of the sound data portion of the WAV file.

Note that header[] is a unsigned char array and the values above are 16 or 32 bit integer values. Also, note that they are Little-Endian in their byte layout. You can construct e.g. sampling rate as:
```
fsamp = header[27]<<24 | header[26]<<16 | header[25]<<8 | header[24];
```

Confirm that the data size it is consistent with the file size from
```
ls –l tone1k.wav
```

3. Binary calculations (20 points)
Use the program in problem (2).

Determine the number of samples in ifile by using the data size and bits per sample from problem (2).

Declare a pointer to short (i.e. 16-bit PCM) to hold all of the tone1k values and allocate storage to the pointer using malloc() with error checking and reporting.
```
short *x;
If ( (x = (short *)malloc(N * sizeof(*x)) == NULL) { … error
```

Read all of the tone1k values into x using fread() with error checking and reporting.

Maximum absolute value can be calculated as

$$vmax = \max\left(\text{abs}(x[i])\right) \; for \; i = 0, \ldots N-1$$

and abs() can be calculated as if .. else statements or as () ? : statement and max() as if () { } statement. The max abs value *vmax* should be type double and set its initial value to DBL_MIN, which requires #include <float.h>.

RMS is calculated as

$$rms = \sqrt{\sum_{i=0}^{N-1} x[i]^2}$$

The variable *rms* should be type double and its initial value is zero. The square root function can be performed by sqrt() which requires #include <math.h>.

Calculate either max abs value or RMS value, depending on which command line arg is present. Print the mode (max abs value or RMS) and the value.

4. Adjust the array x and write to a WAV file (20 points)

Convert the command line arguments v_{dB} from dB to amplitude values, where target_value is type double. You may have to cast v_{dB} to type double so that the operation $v_{dB}/20$ is interpreted as a floating-point division (and not an integer division).

$$target\_value = 10^{(\frac{v_{dB}}{20})}$$

Furthermore, the command-line arg v_{dB} is in units dBFS, were 0 dB is the largest possible value in the data representation. If v_{dB} is 0 dBFS then the forumula above gives *target_value* = 1.0, which is NOT the largest possible value when our data is 16-bit PCM. Hence, add a normalization factor of 32767, which is the largest possible value in 16-bit PCM.

Use the pow() function as

```
target_value = 32767*pow(10, v_dB/20.0);
```

This requires #include <math.h>.

Modify the values of x so that either the max absolute value or RMS value is equal to what is requested in the command line. Do this by multiplying every value of x as

$$xnew = (x)(\frac{target\_value}{x\_value})$$

where *x_value* is the calculated max abs value or rms value using the formulas given above.

Write the adjusted x to ofile with a WAV header. Since the length and format of ofile is identical to that of ifile, you can write out the exact same header that you read in problem (1). Since the header is at the front (head) of the output file, followed by the PCM data, first write the header[] array and then the x[] array.