5.5 Write a function that reads WAV files. WAV is an uncompressed format for sound files that was created by Microsoft. Sound is composed of waves. Digital sound samples the value of these waves at regular intervals. WAV files contain these samples along with metadata describing how they are stored and how quickly they should be played back.

A WAV file starts with an 8 byte header that corresponds to the ASCII characters "RIFFWAVE". After that, it consists of a series of "chunks" Each chunk starts with a "FOURCC" code, which is four bytes, each of which corresponds to a printable ASCII character. This identifies the type of chunk. Next comes a four byte little endian unsigned integer in binary. This gives the number of bytes in the rest of the chunk. To summarize:

field	**bytes**	**type**
FOURCC	4	`char`
size	4	`unsigned int`
contents	size	`unsigned char`

Start by writing a program that reads the header and all of the chunks in a WAV file. Find a WAV file on the internet and use it to test your code. You can tell it works if the FOURCC codes only have printable characters, and you do not reach end-of-file before the end of the last chunk.

Hints:

- You can treat FOURCC codes as strings, but don't forget the null terminator, which the file does not contain.
- You can use the `strncpy` function to compare two strings.
- At this point, read the contents of each chunk as an array of `unsigned chars`.
- Should you create the data array on the stack or on the heap?

Now modify your program to read the contents of the chunks. There are two particular chunks you need to pay attention to. The rest you can ignore. The first is the format chunk, which is identified by a FOURCC code of "fmt ". Note the fourth character is a space. The contents of the format chunk are a number of unsigned integers of different lengths as follows:

name	**bytes**	**comments**
`format`	2	Should be 1, otherwise the data are compressed
`nchannels`	2	1 for mono, 2 for stereo
`rate`	4	The number of samples per second
`byte_rate`	4	`rate*nchannels*bits_per_sample`
`align`	2	`nchannels*bits_per_sample/8`
`bits_per_sample`	2	The number of bits per sample (8 or 16)

You have already read the chunk contents as an array of bytes. To convert these to unsigned integers, cast a pointer to the start of each field to a pointer to the appropriate data type. For example, `(uint16_t*)(contents+2)` gives a pointer to `nchannels`.

The other chunk you need to decode is the data chunk, which is identified by FOURCC code "data". This contains the audio samples stored as signed integers of the size specified in the format chunk. If there is more than one channel, then the samples are interleaved, *e.g.* left channel, right channel, left channel, right channel ... You can use a single pointer cast to turn the `unsigned char` contents array into an array of the appropriate size integer.

Your program should print the sound samples for each channel in a different column. If everything works properly, the numbers should cycle around zero, like waves.

Hints:

- Remember the size-specific integer types, like `uint16_t` and `uint32_t`, are defined in the `cstdint` header file.
- If you find a WAV file where `format` is not 1, then don't try to decode it, go find a different file.
- You can ignore most of the metadata in the format chunk. The important things are `nchannels` and `bits_per_sample`.

Beyond the Question: If you know how to make plots in Excel or another tool, then you can plot the samples *vs.* time to see what they look like. You could also create ASCII art that draws the wave shape with a series of characters scrolling up the screen. If you want an added challenge, you can create a program that writes a WAV file. Make up your own waveform and listen to what it sounds like when you play it.

These days most digital audio is stored in a compressed format, like MP3. MP3 achieves high rates of compression by throwing away information that your ear won't notice. This sort of encoding enabled internet music streaming and sharing in the 1990s. Decoding MP3 files is quite a bit harder.