1. The example in slide 30 assumes mono. Provide an alternative version that will work for stereo. I want only the lines to allocate the array, size it, and load the samples.

```cpp
std::vector<short> wavetable;
wavetable.resize(SampleFrames() * 2);

short audio[2];
for(int i=0; i<SampleFrames(); i++)
{
    ProcessReadFrame(audio);
    wavetable[i*2] = audio[0];
    wavetable[i*2 + 1] = audio[1];
}
```
or:

```cpp
std::vector<short> wavetableL;
std::vector<short> wavetableR;
wavetableL.resize(SampleFrames());
wavetableR.resize(SampleFrames());

short audio[2];
for(int i=0; i<SampleFrames(); i++)
{
    ProcessReadFrame(audio);
    wavetableL[i] = audio[0];
    wavetableR[i] = audio[1];
}
```

2. Indicate changes to the code in slide 31 to accommodate stereo.

```cpp
audio[0] = wavetable[p*2];
audio[1] = wavetable[p*2 + 1];
```
or:

```cpp
audio[0] = wavetableL[p];
audio[1] = wavetableR[p];
```

3. Suppose the current sample is 275.25. Assume sample 275 is 1000 and sample 276 is 1200. What will be the sample at 275.25?
We are 25% of the way from 1000 to 1200, so the sample is:

\[(0.25 \times 1200) + ((1 - 0.25) \times 1000) = 1050\]

4. A violin sample has an attack period of 1000 samples and a loopable period of 3000 samples. Assume we advance the samples like this: frame += framerate; and that frame and framerate are both doubles. Indicate the changes after this line to accommodate looping?

```c
if(frame > 4000)
    frame -= 3000;
```

5. Write code (back) to implement an attack and release where the attack time is in the variable m_attack and the release time is the rest of the audio. Assume we attack to a gain of 1 and decay back to a gain of 0. I want just code to compute the gain at a given point in time. Assume the duration is in the variable m_duration.

```c
double gain;
if(time < m_attack)
    gain = time / m_attack;
else
    gain = 1 - (time - m_attack) / (m_duration - m_attack);
```