1. Suppose you have the equation: \( \sin(\sin(t) t) \). Draw a plot of what you think this equation would generate over time.

What is inside the outer sine is \( \sin(t) t \). \( \sin(t) \) will be a sine wave of amplitude 1. \( \sin(t) t \) will be a sine wave of increasing amplitude. It will be swinging plus and minus larger and larger amounts. See the plot in the lecture notes.

2. Write the code to generate a band limited 1000Hz sawtooth wave. (middle right).

```cpp
for(double time=0.; time < duration; time += 1. / sampleRate)
{
    double s = 0;
    for(int h=1; h<(sampleRate / (2 * 1000)); h++)
        s += sin(2 * M_PI * h * 1000 * t) / h;

    short sample = short(amplitude * s);
    audio[0] = audio[1] = sample;

    GenerateFrame(audio);
}
```

Actual code in AudioProcess:

```cpp
// Call to open the generator output
if(!GenerateBegin())
    return;

short audio[2];

for(double time=0.; time < m_duration; time += 1. / m_sampleRate)
{
    double s = 0;
    for(int h=1; h<(m_sampleRate / (2 * m_freq1)); h++)
        s += sin(2 * M_PI * h * m_freq1 * time) / h;

    audio[0] = audio[1] = short(m_amplitude * s);

    GenerateWriteFrame(audio);

    // The progress control
    if(!GenerateProgress(time / m_duration))
        break;
}
```

// Call to close the generator output
3. Write code that will generate a sine wave sweep from 100Hz to 1000Hz in 10 seconds. (top right)

```cpp
double radians = 0;  // For the sine wave
double samplePeriod = 1. / sampleRate;

for(double time=0.; time < duration; time += samplePeriod)
{
    double freq = 100 + (1000 - 100) * time / 10;

    short sample = short(amplitude * sin(radians));
    audio[0] = audio[1] = sample;

    // Increment the phases
    radians += (2 * M_PI * freq) / sampleRate;

    GenerateFrame(audio);
}
```

4. Write the code for a COscillator::GetSample(). (bottom right).

```cpp
double COscillator::GetSample()
{
    double sample = amplitude * sin(radians);
    radians += (2 * M_PI * m_freq1) / m_sampleRate;
    return sample;
}
```

5. Write the code that will ping-pong a mono mix of an audio between the two audio channels at a rate of 5Hz. (below)

```cpp
// Call to open the processing output
if(!ProcessBegin())
    return;

short audio[2];

double t = 0;
for(int i=0; i<SampleFrames(); i++, t+= 1. / SampleRate())
{
    ProcessReadFrame(audio);
    double mix = 0.5 * audio[0] + 0.5 * audio[1];

    double pongL = sin(2 * M_PI * 5 * t) * 0.5 + 0.5;
    double pongR = 1 - pongL;

    audio[0] = short(audio[0] * pongL);
    audio[1] = short(audio[1] * pongR);

    ProcessWriteFrame(audio);
```
// The progress control
if(!ProcessProgress(double(i) / SampleFrames()))
    break;
}

// Call to close the generator output
ProcessEnd();