This assignment is to make a Pd patch that implements additive synthesis and an ADSR envelope implemented as an abstraction. The sound to be synthesized should be clarinet-like, using theoretically derived harmonics (i.e., every odd harmonic) each with an amplitude that is inversely proportional to its harmonic number (the higher the harmonic, the lower its amplitude).

Consider the following 5 steps when making your patch (each is worth 2 points for a total of 10):

1. Your additive synthesizer should have at least 5 (but no more than 10) sinusoidal oscillators (osc~). The outputs of each oscillator should be multiplied by an ADSR output so that an envelope is applied before being fed into a single multiplier that controls the overall signal gain (feeding multiple signals into a single audio inlet will automatically add them).

2. Use multipliers to calculate the frequency of each osc~ according to some fundamental (sounding) frequency and its harmonic number (i.e. the 3rd harmonic would be 3 times the sounding frequency).

3. Make an ADSR and save as an abstraction. It should take a message as its input (through an inlet), indicating the length of the attack (A), decay (D), sustain (S) and release (R) segments, as well as the sustain level (the attack goes to a level of 1). Within the ADSR, use an unpack object to separate the message so that you have access to each of these values.

4. Within the ADSR abstraction, use a single line~ object for which you create messages implementing each of the ADSR segments. Two delay objects will be needed, one to trigger the decay and one to trigger the release. It may also be useful to use a pack object to concatenate two values into a message, as well as a float object to temporarily hold a value without triggering.

5. Apply an ADSR envelope (each with a different message) to each of your oscillators before summing them to a final output. A general rule of thumb, higher frequencies take longer to reach their steady state and are the first to decay.