I. Preliminary Notes
    A. Much of my testing phase happened during the time I took to complete Development 2. A good amount of testing happened when I showed the code to my professor, Dr. Wells, and we strategized different ways of mitigating certain bugs as I pointed them out to him. Other debugging solutions would come up while developing new aspects, and during the development phase, I would find solutions to those bugs as they arose.

II. Bugs found during Testing
    A. Tempo Slider stopping momentarily while changing the tempo
        1. Issue
            a) One particularly challenging bug was the momentary stoppage of the tempo slider while changing the tempo. As I aimed to replicate the seamless functionality of the original DFAM, I encountered difficulties. Despite numerous attempts and extensive consultations with Chat GPT, the only viable solution I could find was to clear the interval and restart it during the tempo change process. This, unfortunately, led to the sequencer momentarily stopping, creating a jarring experience for the user.
        2. Solution
            a) Instead of having the tempo change actuated when the slider is moved, I changed my code so that the program would check if the tempo had changed during each step. If the tempo changes, the interval can be inconsequentially cleared, and a new interval time can replace it. This allows for no need to pause when the tempo slider is changed, making this a more seamless experience.
    B. At swift tempos, the oscillators' delays are stacked on each other, creating weird sonic effects.
        1. Issue
            a) At tempos higher than 300 bpm, the sound would drastically change due to the ADSR nodes stacking on top of one another. I didn't want this because it was an obvious bug in the software, and the user would be confused about why their sound suddenly shifted.
        2. Solution
            a) I figured this was an issue of how I was increasing the decay. Before, I would just have an addition function using the decay slider value and then use that value to determine how long it would take for the sound to dissipate. I originally made eight different ADSR nodes in the program to create this effect of a longer decay than the step interval because this is how the DFAM functioned. I eventually figured out that because the decay was much longer than the interval duration, at times, the sequence would go through all the steps, and the sound of the first ADSR Node would still be ramping down when it was time for it to start

again (causing out issues). To solve this, I used the interval time and created a multiplication function so that the delay would either divide or multiply the length of the interval duration. This way, at faster tempos, there can only be a 1/10 duration of the interval duration to 4 times the interval duration, effectively solving the issue.

C. The White Noise generator would only play for about 2 seconds and then stop.
    1. Issue
        a) I wanted this white noise generator to run indefinitely so that the user could control and decide if they wanted it in their sound by only controlling the gain of the generator. When I first implemented it, it would only run for about 2 seconds, and then it would go away, only to be brought back by re-initializing the sound.
    2. Solution
        a) I found a way to make the white noise generator run for a much more extended period of time. Somewhere in the set-up code for the generator, I could change a value from 2 to 2000, which made the white noise generator run for a VERY long time. There are issues with this being that when the user hits the initialize sound button, there is about a 1-second buffer due to this variable's high processing power. Still, I figured it was okay because, at this point, the user could now use the white noise generator for an extremely long period of time without worrying about it going away.

D. Step Pitch changes, taking the oscillators out of their intervals.
    1. Issue
        a) When one changes the step pitch, it changes the pitch of oscillators 1 and 2. If, for instance, oscillator 1 was at 110hz, oscillator 2 was at 220hz, and the step pitch slider was up, it would add about 100 hz to each oscillator. This would take the octave we originally had and change the interval, creating a drastically different sound.
    2. Solution
        a) To solve this, I simply made it so the step pitch slider would multiply or divide the frequencies in increments of 0.1. This way, when oscillator 1 is at 220, and oscillator 2 is at 440, increasing the step pitch slider would change it so that oscillator 1’s frequency changes to 440 and oscillator 2’s to 880, which is still an octave. This means that the relationship between the two oscillators can only be changed when the frequency slider for each is changed (which is as intended)