I. Purpose

For the purpose of increasing my knowledge of and in audio programming. I plan to create and design an audio plugin that will run as a virtual synthesizer and sampler in a host software of any digital audio workstation program. The audio plugin will use digital processing to simulate analog synthesizers in software

II. End Product

The audio plugin will feature:

- Process audio
- Process midi data
- Graphical user interface
    - Display current sound being used
    - Knobs (parameters) that will manipulate the audio being processed
    - List of sounds that can be created and saved

III. Central Skills/Practice

- JUCE
- C++
- Pure data
- Digital Signal Processing
- DAW workstation

IV. Basic work schedule (15 weeks)

Week 1:
-- Installing xcode

-- Importing the WDL-ol c++ library

-- Create a basic GUI design with two knobs that will control the volume and reverb effect

-- Checkpoint: Audio plugin has a simple GUI layout with master volume and another knob the reverb effect.

Week 2:
-- improve GUI design
-- Create an Oscillator that will generate sine, saw, square, and triangle waveforms
-- Create a function for applying these waveforms
-- Checkpoint: The GUI should have a more complex design and can generate an

oscillating waveform. Note: The waveform frequency shouldn't pass a certain range of Hertz

Week 3/4:
-- Creating a midi data receiver that will take midi inputs from the DAW program and midi devices. The midi input will pitch the output depending on the note the user is pressing.
-- Checkpoint: The audio plugin can receive midi data played through an external midi device and can be manipulated it within the audio plugin.

Week 5:
-- Implement envelopes to manipulate the stages of attack, decay, sustain and release of any sound
-- Checkpoint: The audio plugin will be able to determine the velocity of the note being note pressed

Week 6:
-- Implement triggers for the stages of attack, decay, sustain and release so when the note is pressed or not being pressed the envelop is in the stage of attack, and when is released, the envelop switches to release
-- Checkpoint: The audio plugin will determine how the volume of each note is affected such as how hard each note is hit, how long that the note is sustained when being pressed and how long that note will resonate after being released.

Week 7:
-- Implement envelope GUI and create knobs that controls the envelope stages
-- Checkpoint: The envelope GUI should be able to control attack, decay, sustain and release

Week 8:
-- Implement an equalization (EQ) that will cut out low, mid and high frequencies. The EQ will also have a GUI
-- Checkpoint: The audio plugin will allow edit the sound by cutting or adding to a specified frequency

Week 9/10:
-- Implement resonance that will develop the peak of the cutoff frequency for the filter.
-- Create a filter envelope that acts as the opacity of the filter through the changes in the stages of the envelope
-- Checkpoint: The resonance, and filter envelop knobs work correctly as intended

Week 11:
-- Implement low frequency oscillator that can produce the same waveforms as the last oscillator but at lower frequency
-- Redesign GUI because the aspects of synthesizers are mostly developed.

-- Checkpoint: The Low frequency oscillator works, and GUI design is more complex

Week 12:
-- Implement polyphony audio processing which allows for multiple notes to be played at once rather than one.
-- Checkpoint: polyphonic works

Week 13-15:
-- Implement arpeggiator which allows a step through a sequence of notes based on midi input, most often from a keyboard MIDI controller
-- Implement pitch modulation which raise or lower transposition of a sound.
--Debug any faults and export
-- Give it to music producers to test

-- Checkpoint: Finished project