**Test Plan:** Describe what testing framework(s) you're using and what parts of your codebase you're concentrating on testing, with a brief explanation justifying that plan.

Since we have a C++ project, it would make sense to use the Google testing library covered in class for unit testing. Our code includes drop-in replacements for many standard functions (strlen, memcpy, strings, double parsing/serializing, etc), so it would make sense to unit test these functions. Additionally, we have a custom expression parser and interpreter which would also be good to unit test. Beyond these subsections however, our code becomes very hard to unit test due to the functionality of the code and the nature of its inputs and outputs (UI code and audio processing). Thus we have decided to concentrate a substantial amount of our testing efforts on manual testing which is more conducive to ensuring our application has the overall intended functionality and resilience we desire.

**Test Assessment:** Describe how you plan to assess the quality of your tests. For example, which tools are you using to evaluate coverage (or other measurements of test quality)?

For unit testing, using metrics such as code coverage (especially branch coverage) as well as a general volume of tests will be the best way to measure the quality of our tests. For subjective testing there is less of an objective metric for assessing quality, however, we are striving to produce a large volume of tests that assess the behavior of the program both performing actions that a typical user would perform as well as well as intentionally looking for and attempting to exploit edge cases (such as unreasonably large graphs and loading corrupted audio files). Intuitively, we feel that crafting subjective tests in this intentional and targeted manner will yield the highest quality tests that will tell us the most about the state of our application.

**Current State:** Summarize your testing progress so far, according to the test plan. How many tests do you have? What's the quality based on your test assessment?

- There is a subjective testing document designed to test the software as a whole, configuring the UI to see how it interacts with other elements.
    - All tests combined currently evaluate the following (14 finished, at least 21 in the end):
        - Node interactions/scalability
        - UI features working correctly
        - Error handling and recovery
        - Real-time/dynamic performance
        - Edge Cases and scenarios not often encountered
        - Audio output
- There are 20 tests so far for the units of code, but many more can be created with the same framework. It's hard to use an external tool with the custom inclusion of libraries in this project, but the same fundamentals are being followed here. Each function in the

StrA class has a happy and a sad test, with some containing more than one. This only applies to the unit tests of our custom standard library implementations.