SMU | LYLE
SCHOOL OF ENGINEERING

CS 7345 ADVANCED APPLICATIONS: LAB 2

Overview: In this lab you will utilize Emscripten to convert a c++ library/application into a JS library via WebIDL and WebAssembly (WASM) and build a demo application. After conversion, evaluate the library performance.

Due: October 19, 2022 @ 11:59pm.

Code Requirements:

1. Identify code base to convert to a JS library .
    a. Any code base can be used that meets the requirements below
        i. Preferred code base should be related to your research or personal work, but that isn't a requirement, an example project will be provided to convert as a backup if nothing else can be found.
        ii. An existing c++ code base can be used that is publicly available and doesn't currently have an emscripten port library port
        iii. An existing custom library/program you have previously created (algorithms, search, math lib, etc) can be used, but all code must be included with lab submission.
        iv. No multithreading is required for this lab but will be added in future labs. Picking a project that is capable of multithreaded execution would be beneficial for future labs.
2. Create a well-defined API for the library
    a. Interface should be well designed for usability and efficiency
3. Create a test application to demonstrate your JS library (demo application).
    a. Your application can either run in NodeJS or in a browser.
    b. Application should test all features of your library and provide feedback to user on each feature and its use.
    c. Application can be designed as either a technology demo or a product demo for your library (depends on selected code base). It should fully demonstrate and make all features understood to user from its operation.
    d. Create ability for application to output performance data
4. Create comparison version for test application (comparison app).
    a. This application can be a native JS or C++ application.
        i. You can select which language will work better for your comparison.
        ii. You only need 1 language to be used for comparison, not both.
    b. It will be the comparative performance measure for the transpiled library
    c. Create ability for application to output performance data.
5. Extra Credit: Attempt to improve performance of WASM code, if possible, by modifying native and/or JavaScript code

<u>Report Requirements:</u>

1. Document API for library
    a. Provide developer documentation for the API explaining each endpoints use and functionality. Be sure to explain fully the input/output for method, expected outcomes as well as any assumptions that are made about its use and/or data requirements.
    b. Provide a second section that describes the design concepts on how the library was created. Be sure to describe in detail any design patterns, class structure and provide explanations, pro/cons for design decisions and implementations.
    c. High-level class layout/UML should be included in writeup
2. Compare and Contrast execution time of the library code between demo application and comparison application
    a. Perform a timing analysis for each code base
        i. Your analysis should be performed over multiple executions of the code base
            1. Do not compare off a single execution of each code base
        ii. Show confidence intervals for execution time at 95% confidence interval and discuss if the results are statistically significant
    b. Extra Credit: perform same analysis above on any optimizations performed on code base.

Submission should reside in the repo with given folder structure, with all code in the "Code" folder, raw data files generated in "Data" and your final report in "Report" folder in the git repo.