CS 3341 FOUNDATIONS OF COMPUTING: LAB 3 FLOWSCRIPT INTERPRETER

Overview: Design and build a FlowScript Interpreter and execute FlowScripts via Multithreaded Job System library.

Repo Link:

Due: November 13, 2023 @ 11:59pm.

Code Requirements (80%):

- Build upon the JobSystem and FlowScript language developed in Lab 2. You may modify, change, update any portion of previous labs to meet the objectives defined in Lab 3. JobSystem must be loaded as a library as in Lab 2.
- The interpreter may take 1 of 2 approaches:
    - Interpret and execute individual logical units of a script one at a time, and repeat until end of script is reached
        - This approach models a "line by line" interpreter, executing each command (logical unit) of script as it is interpreted.
        - Slower executional approach, but more flexibility on FlowScript execution logic
    - Interpret entire script, and create an executional chain that will then be processed in its entirety without any intermediate interpretation steps needed.
        - This approach interprets the entire script, and translates entire script into executable code.
        - Faster execution, as all interpretation is done at beginning and output is a final chain of commands of executable code that can be re-run without any need for reinterpretation.
        - Requires the entire logic of FlowScript to be recreated/modeled into the executable code chain.
- Create a FlowScript Interpreter that can perform the following actions:
    - Lexical Parsing
        - The interpreter should be able to take either a portion or entire FlowScript and segment it into the lexical tokens needed for interpreter to perform sematic and syntactical meaning of the code provided.

    - Syntactic Parsing
        - The interpreter should be able to parse the lexical tokens to create an executable code. This can be done via either of the interpreter approaches described above
    - Both Lexical and Syntatic errors in FlowScripts should be reported back to the user with debug information to provide user insight on why the FlowScript failed either of the parsing processes.
    - The interpreter can be coded in any language, but the execution of jobs/process modeled in the FlowScripts, must be executed in c++ JobSystem. You are free to call the JobSystem from another language/platform using FFI or something similar.
- Create a test program that will demonstrate the FlowScript interpreter capability to parse FlowScripts.
    - Ensure that all the features of FlowScript are demonstrated in the test program and the scripts it is executing.
    - The test program should load FlowScripts from a file, and not be hard coded text within your project.
- Extra Credit: Enable the ability for FlowScript to go multifile. This should allow a hierarchical approach to process/job definitions. Where a single file that contains multiple processes, can be added to a process flow in another files as a single process element. Your final project must be able to execute and interpret the multifile system to receive credit.
- ***CS 5393 Requirement***: Make the Lexical and Syntactic Parsing run as a job/thread within the JobSystem. Utilizing the custom job interface created in Lab 2.

Report Requirements (20%):

- Document and describe the rules utilized for both Lexical and Syntactic parsing.
    - You can utilize any methodology you want to create the rules, BNF is one possible approach by not a requirement.
    - The rules should account for all features defined in the FlowScript language
- Provide developer documentation on error types.
    - Each lexical and syntactical error type should be documented in detail to help users understand the error type
    - Examples should be shown for each error type, and example errors shown should also include a corrected version so user can see how fixes were made to remove error.

All code in the "Code" folder, raw data files generated in "Data" and your final report in "Report" folder in the git repo. If your project fails the compile action upon final push to git repo, it will receive a 0. Be sure to resolve any issues with the compile action prior to deadline.