**Ma. Adriana Nicole T. Trinidad BSCS 3A**

Working on the LOLCODE interpreter project from start to finish was both a fun and educational experience. At the beginning, I didn’t know how interpreters function internally, but this activity gave me a step-by-step understanding from tokenizing raw code to evaluating and executing each command. It was interesting to see how a human-readable language like LOLCODE can be converted into logic that a machine can follow.

In Week 1, we started with researching LOLCODE syntax and features. I learned about how it handles variable declarations, input/output with VISIBLE and GIMMEH, and conditionals like O RLY? and YA RLY. We also explored how an interpreter is structured, including the Lexer, Parser, and Evaluator components. Our team set up the GitHub repository, added our initial research, and created a flowchart explaining how the interpreter would process LOLCODE programs.

In Week 2, we began writing the actual Python code. I contributed to testing the interpreter, making sure it could handle inputs correctly and debugging issues with variables and expressions. We created the main interpreter files: lexer.py for breaking down the code, parser.py for building syntax trees, and interpreter.py for executing the logic. We made two test files: hello\_world.lol and conditional.lol, and ran them through our interpreter to verify the outputs.

Week 3 was focused on enhancements and completing the final product. We added features like SMOOSH for string concatenation, TROOF values (WIN/FAIL), nested expressions, and proper error messages with line numbers. We also built a GUI using Tkinter so users can run LOLCODE files without using the command line. I helped test each .lol file thoroughly to make sure everything still worked, even with the added complexity.

One major challenge I faced was formatting user input and making sure printed outputs appeared correctly. It was tricky handling string outputs using SMOOSH and keeping the expressions readable. Another difficulty was integrating GUI input/output since our code was originally built for terminal use.

Despite the challenges, I enjoyed seeing everything come together, from research and design to a fully working interpreter. I learned a lot about language parsing, debugging, and teamwork. This project gave me confidence to work on more complex coding tasks in the future.