Example Project Proposal for Example Project Report

Student Name(s): Kyle Dewey
Proposed System Under Test (SUT): calculator
Link to SUT Source Code: **https://github.com/CSUN-COMP587-F18/calculator**
SUT Size: 427 Lines of Code

SUT Description

A calculator that takes a command-line argument, representing an arithmetic expression. Tokenizes, parses, and evaluates the expression, and displays the result.

Attributes

- **Simple**: The calculator has a simple, intuitive interface
- **Safe:** The calculator safely handles invalid input, and will not crash.
- **Correct**: The calculator calculates the correct answer given a valid input.

Components

- **Lexer**: Breaks down expressions into a sequence of tokens
- **Parser**: Converts sequences of tokens into abstract syntax trees representing arithmetic expressions
- **Interpreter:** Evaluates arithmetic expressions

Capabilities:

- **Lexer** is **Safe**: The lexer never crashes, and delivers informative error messages on invalid input.
- **Lexer** is **Correct**: The lexer correctly tokenizes valid input, yielding tokens.
- **Parser** is **Safe**: The parser never crashes, and delivers informative error messages on invalid input.
- **Parser** is **Correct**: The parser correctly parses valid input, yielding an abstract syntax tree.
- **Interpreter** is **Safe**: The interpreter never crashes, and delivers informative error messages on invalid input (e.g., division by zero).
- **Interpreter** is **Correct**: The interpreter gives correct answers for valid input.

Capabilities Count:

	Simple	Safe	Correct
Lexer	0	1	1
Parser	0	1	1
Interpreter	0	1	1

Basic Testing:
For each one of the listed capabilities above, I plan to write unit tests. There are currently only unit tests for part of a common library.

Advanced V&V:
I plan to write fuzzers to automatically test each one of these components, and ensure they never crash. Specifically:
- **Lexer**: a fuzzer to automatically generate different character sequences.
- **Parser**: a fuzzer to automatically generate different token sequences.
- **Interpreter**: a fuzzer to automatically generate different abstract syntax trees.