Assignment 0 – LRC Report Template

Nickolas Tran

CSE 13S – Winter 24

Purpose

The purpose of testing the behavior of the program when given more than two arguments is to ensure that the program behaves as expected according to the chosen interpretation. Given the ambiguity in the specifications regarding the behavior of calc in such situations, testing this scenario helps to:

- Determine how the program behaves when provided with extra arguments.
- Ensure that the program behaves according to the chosen interpretation, whether it ignores extra arguments and produces a result based on the first two arguments or produces an error message and returns a non-zero exit code.
- Validate that the program handles unexpected input correctly and provides appropriate feedback to the user.

Problem 1

Describe a test plan for the various implementations of calc. Given that the program cannot be tested on every possible input, what are three examples of tests that are implied by the spec but not checked bybasic arithmetic.sh?
To ensure the various implementations of calc meet the specifications, a comprehensive test plan must be devised. Given that testing every possible input is impractical, we need to focus on testing representative cases.
Test Cases:
Basic Arithmetic Tests:

- Test the basic arithmetic operations: addition, subtraction, multiplication, and division.
- Ensure that the calculator produces the correct result for simple arithmetic expressions.

Edge Cases:

- Test the calculator with extreme values such as very large or very small numbers.
- Test with zero, as this may have special behavior in division.

Invalid Input Handling:

- Test the calculator's behavior when invalid input is provided (e.g., non-numeric input).
- Ensure that the calculator handles errors gracefully and provides informative error messages.

Expression Parsing:

- Test the calculator with complex arithmetic expressions containing multiple operators and parentheses.
- Ensure that the calculator follows the correct order of operations.

Examples of tests implied by the spec but not checked by basic arithmetic.sh:

Division by Zero:

- Test the calculator's behavior when dividing by zero. It should produce an error message and return a non-zero exit code.

Handling of Extra Arguments:

- Test the calculator's behavior when provided with more than two arguments at the command line.
- It should either ignore extra arguments and produce the result based on the first two arguments, or produce an error message and return a non-zero exit code, depending on the interpretation chosen.

Precision and Rounding:

- Test the calculator's handling of floating-point arithmetic.
- Ensure that the calculator produces accurate results and handles rounding appropriately.

Problem 2

You may have noticed that the spec doesn't say anything about what to do if the user supplies too many arguments at the command line. Unfortunately, ambiguity in specifications is very common in practice and can be very confusing for programmers. If you were asked to implement calc given this spec, you would be forced to choose from among at least three interpretations of what calc should do when called with, say, three arguments 3, 4, and 5:

- Calc should sum *all* of the arguments, and print 12.
- Calc should ignore the third argument, and print the sum of the first two: 7.
- Calc should print an error message and return non-zero.

Luckily for you, you are not the programmer (for now). Should your test scripts check what the program does when given more than two arguments? Why or why not?
Yes, the test scripts should check what the program does when given more than two arguments. Given the ambiguity in the specifications regarding the behavior of calc when supplied with more than two arguments, testing this scenario is crucial to ensure that the program behaves as expected. Depending on the chosen interpretation, the program should either ignore extra arguments and produce the result based on the first two arguments or produce an error message and return a non-zero exit code. Therefore, testing this scenario will help verify that the program behaves according to the chosen interpretation.