Software Engineering Team Project: Deliverable #5

November 21, 2017

Team Members

Joe Ayers
Maz Little
Jasmine Mai
Joe Spencer

Introduction

The purpose of Deliverable 5 is to design and inject 5 faults into the code being tested that will cause at least 5, but hopefully not all, tests to fail.

Process

As stated in our last report, we decided to move on from testing the incredibly simple functions that sugar uses to just call Python math library functions, to more complicated logic utilized by the sugar OS. These included functions such as factorize, binary conversion, and determining if a number is an integer. However, when we went into the code again to assess it for possible fault injection sites, we found that the logic still wasn't complicated enough for an injected fault to feel like it actually had an impact. To that end, we modified and added to our test case list, so that we now are testing:

- Pow
    - A function to raise a specified number to a specified power
    - The function included a line with a double asterisk for multiplication; we removed one asterisk at line 418
    - We already have test cases for this function
- Factorize
    - A function to find the prime factors of a number
    - The function included a line where the number passed to it was divided modularly; we changed this modular division to regular division at line 291
    - We already have test cases for this function
- Factorial
    - A function to calculate the factorial of an integer
    - We removed the check to make sure the number is less than 0 at line 261, and inserted a test case for a negative number.
    - Test cases:
        - Null
        - Empty
        - 0

        - -1
        - 4
- Inverse function
    - A function to invert a number passed to it. I.e., if the function is passed '4,' it returns ¼
    - We edited the comparison to assert whether or not the numerator of the number passed to this function is 0 (as if it is, the inverse will be impossible). To that end, we changed the comparison to another, wrong comparison (== to >) at line 319
    - Test cases
        - Null
        - Empty
        - 0
        - Abc
        - 0.5
        - 25
- B10bin
    - A function to convert a decimal number to a binary number
    - In the function, there is a comparison to make sure that the value passed is bigger than 0. We changed the comparison from >= to > on line 188
    - We already have test cases for this function

Report

It was relatively easy for us to assess the functions we selected for our test cases, and determine the best places to insert faults. We tried to approach fault insertion by making changes that could conceivably be the result of human error, rather than trying to come up with large, structural errors. It helped that the functions we are testing fall on the simpler side. At this point in the project, the code and script are familiar enough to us that this was a relatively simple task to fulfill after walking through the code in a group.