SWE 30009 Software Testing & Reliability TP 2 2024
Title: Assignment 1

Name (Student ID): Joseph Linao (104556329)

Lab class: Wednesday / 10:30 AM to 11:30 AM / BA709

Due Date: Friday 31st October 2024 at 11:59 PM

Task 1:

Program breakdown:

Input A, B // A and B are integer variables, so we input A and B

Operation 1: A = (A + B) * B // input A equals to the bracket of the addition of two variables, then multiplied by B

Operation 2: C = A – 5 // input C is equal to the variable A subtracted by 5

Output C // C is an integer variable, so it outputs C

Test cases:

Addition based test cases:

- 1: A = (A + B) + B, C = A + 5
- 2: A = (A +B) + B, C = A – 5
- 3: A = (A + B) + B, C = A * 5
- 4: A = (A + B) - B, C = A + 5
- 5: A = (A + B) - B, C = A - 5
- 6: A = (A +B) - B, C = A * 5
- 7: A = (A + B) * B, C = A + 5
- 8: A = (A + B) * B, C = A – 5
- 9: A = (A + B) * B, C = A + 5

Subtraction based test cases:

- 10: A = (A - B) + B, C = A + 5
- 11: A = (A - B) + B, C = A – 5
- 12: A = (A - B) + B, C = A * 5
- 13: A = (A - B) - B, C = A + 5
- 14: A = (A - B) - B, C = A - 5
- 15: A = (A - B) - B, C = A * 5
- 16: A = (A - B) * B, C = A + 5
- 17: A = (A - B) * B, C = A – 5
- 18: A = (A - B) * B, C = A + 5

Multiplication based test cases:

- 19: A = (A * B) + B, C = A + 5

- 20: A = (A * B) + B, C = A – 5
- 21: A = (A * B) + B, C = A * 5
- 22: A = (A * B) - B, C = A + 5
- 23: A = (A * B) - B, C = A - 5
- 24: A = (A * B) - B, C = A * 5
- 25: A = (A * B) * B, C = A + 5
- 26: A = (A * B) * B, C = A – 5
- 27: A = (A * B) * B, C = A + 5

How to get these test cases:

- Based on the original program
- Testing on multiple scenarios by swapping different arithmetic operators presented in the original program (+, -, *)
- These can help in identifying faults related to the incorrect use of the arithmetic operators
- The goal is identifying which test cases can detect incorrect use of the arithmetic operators

Testing objective:

- To find incorrect use of the arithmetic operators such as +, -, or *

Justification:

- Test case 7: detects an error if C uses addition instead of subtraction
- Test case 8: it's the original
- Test case 14: detects an error in both subtraction in A=(A-B) -B
- Test case 21: detects an error when using multiplication instead of subtraction for 'C' and the incorrect operation for 'A'

Task 2:

Review for test cases (using (10, 0)):

- Test case 10: A = (10 - 0) + 0 = 10, C = 10 + 5 = 15
- Test case 14: A = (10 -0) – 0 = 10, C = 10 – 5 = 5
- Test case 21: A = (10 * 0) * 0 = 0, C = 0 – 5 = -5

Justification:

- Since b=0, these test cases ensures that faults involving addition, subtraction, and multiplication are effectively detected.

Task 3:

Concrete testcases:

- Test case 10: A = (A - B) + B, C = A + 5
- Test case 8: A = (A + B) – B, C = A – 5
- Test case 14: A = (A – B) – B, C = A – 5
- Test case 21: A = (A * B) – B, C = A * 5

All selected test cases are concrete test cases as they can correctly execute without faults especially using the inputs A = 10, and B = 0, confirming that no operator faults are present in these cases.

Task 4:

The test cases 10, 8, 14, and 21 were effective in detecting operator faults, especially considering scenarios if one of the inputs is a zero, such as B=0. The coverage of arithmetic operators was robust, though adding a few more test cases could further enhance the fault detection capabilities.