SWE30009 – Software Testing and Reliability

Assignment 1 – Do Duy Hung – 104182779

Task 1: In this task 1, I am allowed to divide it into 3 cases to design the test cases: 2 cases where one arithmetic operator is incorrect and 1 case where both arithmetic operators in both expressions are incorrect.

1, In the case where the "-" operator in the expression A = A - B is incorrect, while the "*" operator in the expression C = A * 2 is correct and the output is C, there are three test cases: A = A + B, A = A * B, and A = A / B. In total, there are 3 test cases for this situation.

2, In the case where the "-" operator in the expression A = A - B is correct, while the "*" operator in the expression C = A * 2 is incorrect and the output is C, there are three test cases: C = A + 2, C = A - 2, and C = A / 2. In total, there are 3 test cases for this situation.

3, In the case where both the "-" operator in the expression A = A - B and the "*" operator in the expression C = A * 2 are incorrect, and the output is C, there are nine test cases. These include by the table I have illustrated below.

First expression	Second expression
A = A + B	C = A + 2
A = A + B	C = A - 2
A = A + B	C = A / 2
A = A * B	C = A + 2
A = A * B	C = A - 2
A = A * B	C = A / 2
A = A / B	C = A + 2
A = A / B	C = A - 2
A = A / B	C = A / 2

In conclusion, the program has a total of 15 possible cases where failures could occur.

Task 2: To determine whether the test case A = 3 and B = 1 is feasible and should be used for testing, it must be ensured that the result of substituting the values of A and B into the correct and incorrect expressions is different. If the results are the same, then it is not a reliable test case.

1, In the first arithmetic operators “A = A - B”, this is the correct usage, we have to compare the results with 3 wrong test case that I have mentioned in task 1.

Correct: A = A – B = 3 – 1 = 2 => A = 2

Incorrect:

- A = A + B = 3 + 1 = 4 => A = 4
- A = A * B = 3 * 1 = 3 => A = 3
- A = A / B = 3 / 1 = 3 => A = 3

⇨ It can be seen that the result between the correct calculation and the incorrect calculation is different. In this case, the values A = 3 and B = 1 can definitely be used as a reliable test case.

2, In the second arithmetic operators "C = A * 2", this is the correct usage, we have to compare the results with 3 wrong test case that I have mentioned in task 1. Note that the value of A in this second expression is the result of the first calculation, meaning A = 2.

Correct: C = A * 2 = 2 * 2 = 4

Incorrect:

- C = A + 2 = 2 + 2 = 4
- C = A – 2 = 2 – 2 = 0
- C = A / 2 = 2 / 2 = 1

⇨ It can be seen that the result between the correct calculation and the first incorrect calculation is the same. In this case, the value A = 3 and B = 1 cannot definitely be used as a reliable test case.

In conclusion, test case A = 3 and B = 1 is not able to achieve the required testing objective.

Task 3:

As I mentioned in the previous two tasks, to find concrete test cases, it is important to ensure that the results obtained from the correct and incorrect calculations are different. While it is possible to manually search for concrete test cases, this would be time-consuming. Instead, I used Python to programmatically identify reliable test cases. I created three definitions: one *`def`* for the correct expression as required by the problem, one *`def`* for incorrect expressions with three test case design scenarios from Task 1 (totaling 15 cases), and the final *`def`* to fulfill the problem's requirement of finding values where the result of the correct expression does not match any of the incorrect ones. Here, I take values of A and B from the first 4 natrual numbers (0 to 3) for easier visualization and calculation.

```
def correct_expression(A, B):
    return (A - B) * 2

def generate_incorrect_operators(A, B):
    wrong_outputs = []

    # I will consider three cases:

    # 1, The "-" operator in the expression A = A - B is incorrect, while the "*"
operator in the expression C = A * 2 is correct, and the output is C.
    wrong_outputs.extend([(A + B) * 2,
                          (A * B) * 2])
    if B != 0:
        wrong_outputs.append((A / B) * 2)

    # 2, The "-" operator in the expression A = A - B is correct, while the "*"
operator in the expression C = A * 2 is incorrect, and the output is C
    wrong_outputs.extend([(A - B) + 2,
                          (A - B) - 2,
```

```
                                (A - B) / 2])

    # 3, All the "-" operator in the expression A = A - B and the "*" operator in
the expression C = A * 2 are incorrect, and the output is C.
    wrong_outputs.extend([(A + B) + 2,
                                (A + B) - 2,
                                (A + B) / 2,
                                (A * B) + 2,
                                (A * B) - 2,
                                (A * B) / 2])
    if B != 0:
        wrong_outputs.extend([(A / B) + 2,
                                    (A / B) - 2,
                                    (A / B) / 2])

    return wrong_outputs

def task3():
    task3_outputs = [
        {"A": A, "B": B}
        for A in range(4)
        for B in range(4)
        if correct_expression(A, B) not in generate_incorrect_operators(A, B)
    ]

    print(task3_outputs)

task3()
```

Output: [{'A': 0, 'B': 3}, {'A': 1, 'B': 2}, {'A': 2, 'B': 1}, {'A': 2, 'B': 3}, {'A': 3, 'B': 2}]

Note that I only use numbers from 0 to 3, so there will be many other concrete test cases with higher or lower values.

Task 4:

Similarly to task 3, I will also create an additional function for task 4. With B = 1 as provided in the problem, I will set `B in range(1, 2)` because this function will only take B = 1. Opposite with task 3, we must find the concrete test cases which cannot achieve the testing objectives, so the concrete test cases will be included in `def `correct_expression and in `def `generate_incorrect_operators. The value of A will temporarily be set to the first 10 values, from 0 to 9.

```
def task4():
    definitions_output = [
        {"A": A, "B": B}
        for A in range(10)
        for B in range(1,2)
        if correct_expression(A, B) in generate_incorrect_operators(A, B)
    ]

    print(definitions_output)

task4()
```

Output: [{'A': 0, 'B': 1}, {'A': 1, 'B': 1}, {'A': 3, 'B': 1}, {'A': 4, 'B': 1}, {'A': 5, 'B': 1}]

Or I can try with nagative integer. I will set A in range (-7, 0)

```
def task4():
    definitions_output = [
        {"A": A, "B": B}
        for A in range(-7,0)
        for B in range(1,2)
        if correct_expression(A, B) in generate_incorrect_operators(A, B)
    ]

    print(definitions_output)

task4()
```

Output: [{'A': -1, 'B': 1}]

Note that I only use numbers from -7 to -1, so there will be many other concrete test cases with higher or lower values.