Name: Trac Duc Anh Luong - ID: 103488117

Assignment 1

Task 1

To design the test cases, we need to consider all possible incorrect usages of the given arithmetic operators. There are 3 possible incorrect usages:

1. "-" in the statement of "A = A - B" is incorrect and "*" in the statement of "C = A * 2" is correct
   a. A = A + B
   b. A = A * B
   c. A = A / B
2. "*" in the statement of "C = A * 2" is incorrect and "-" in the statement of "A = A - B" is correct
   a. C = A + 2
   b. C = A - 2
   c. C = A / 2
3. Both "-" and "*" are incorrect
   a. A = A + B and C = A + 2
   b. A = A + B and C = A - 2
   c. A = A + B and C = A / 2
   d. A = A * B and C = A + 2
   e. A = A * B and C = A - 2
   f. A = A * B and C = A / 2
   g. A = A / B and C = A + 2
   h. A = A / B and C = A - 2
   i. A = A / B and C = A / 2

⇒ There are a total of 15 cases where the program may introduce failures.

Task 2

Suppose we use test case (A=3, B=1) to test the above program:
Consider the 1st arithmetic operator:
Correct usage:

a. A = A - B = 3 - 1 = 2

"-" in the statement of "A = A - B" is incorrect and "*" in the statement of "C = A * 2" is correct:

b. A = A + B = 3 + 1 = 4
c. A = A * B = 3 * 1 = 3
d. A = A / B = 3 / 1 = 3

⇒ *In the 1st arithmetic operator, the correct usage result is different from the incorrect usage.*
Consider the 2nd arithmetic operator:
Correct usage:

a. C = A * 2 = 2 * 2 = 4

"*" in the statement of "C = A * 2" is incorrect and "-" in the statement of "A = A - B" is correct:

b. C = A + 2 = 2 + 2 = 4

c. C = A - 2 = 2 - 2 = 0

d. C = A / 2 = 2 / 2 = 1

⇒ *In the 2nd arithmetic operator, there is one incorrect usage that produces the same result as the correct usage: C = A + 2 = 2 + 2 = 4.*

Therefore, the given test case (A=3, B=1) **cannot achieve the required testing objective**.

Task 3

Based on our design in Task 1, we can write a program to generate "concrete test cases" that can achieve the required testing objective. Concrete test cases are pair values of A and B that are fault-causing inputs. For this task, I have written a Python program with the functions correct_program(A, B) and incorrect_programs(A, B). Then, we can run 2 nested for loops for the values of A and B to check if the returned value of the correct input is in the returned array of incorrect inputs. If not, we can append the value of A and B to an array and return it once the function ends.

Program:

```
def correct_program(A, B):
    A = A - B
    C = A * 2
    return C

def incorrect_programs(A, B):
    incorrect_outputs = []

    # Scenario 1: "-" operator is incorrect, "*" operator is correct
    incorrect_outputs.append((A + B) * 2)
    incorrect_outputs.append((A * B) * 2)
    if (B != 0): # constraint: divisor cannot be 0
        incorrect_outputs.append((A / B) * 2)
    # Scenario 2: "-" operator is correct, "*" operator is incorrect
    incorrect_outputs.append((A - B) + 2)
    incorrect_outputs.append((A - B) - 2)
    incorrect_outputs.append((A - B) / 2)
    # Scenario 3: Both "-" and "*" operators are incorrect
    incorrect_outputs.append((A + B) + 2)
    incorrect_outputs.append((A + B) - 2)
    incorrect_outputs.append((A + B) / 2)
    incorrect_outputs.append((A * B) + 2)
    incorrect_outputs.append((A * B) - 2)
    incorrect_outputs.append((A * B) / 2)
    if (B != 0): # constraint: divisor cannot be 0
        incorrect_outputs.append((A / B) + 2)
        incorrect_outputs.append((A / B) - 2)
        incorrect_outputs.append((A / B) / 2)

    return incorrect_outputs
```

```
def task3():
    distinct_outputs = []

    for A in range(0, 5):
        for B in range(0, 5):
            if correct_program(A, B) not in incorrect_programs(A, B):
                distinct_outputs.append({"A=" + str(A), "B=" + str(B)})

    print(distinct_outputs)

task3()
```

Output:

```
[{'A=0', 'B=3'}, {'B=4', 'A=0'}, {'A=1', 'B=2'}, {'A=1', 'B=4'}, {'B=1',
'A=2'}, {'A=2', 'B=3'}, {'B=2', 'A=3'}, {'B=4', 'A=3'}, {'A=4', 'B=3'}]
```

Note: We can expand the range of A and B to generate more test cases. However, for this example, I am only using values from 0 to 5 inclusively.

Task 4

Given B=1, we can find all possible values of A so that the concrete test cases (A,B) cannot achieve the above testing objective, leveraging the code from task 3. Conversely from Task 4, we need a function that loops through all possible values of As (in the following example, I am setting the range of A from -1000000 to 1000000) and check if the returned value of the correct input is in the returned array of incorrect inputs, if yes, we can append the value of A to an array and return it once the function ends.

Program:

```
def task4():
    similar_outputs = []

    # Given B = 1 as Task 4 requirement:
    B = 1

    for A in range(-1000000, 1000000):
        # append A if the correct program's output is equal to the incorrect
program's output
        if correct_program(A, B) in incorrect_programs(A, B):
            similar_outputs.append(A)

    print(similar_outputs)

task4()
```

Output:

```
[-1, 0, 1, 3, 4, 5]
```