SWE30009 – Software Testing and Reliability

Assignment 1

Student name: Vi Luan Dang

Student ID: 103802759

Task 1:

According to the given program there are 4 feasible arithmetic operators for each operation, therefore, there would be a total of 16 possible feasible arithmetic operator combinations. Among those combinations, there would be 1 correct combination and the rest would be incorrect. This can be visually represented below:

	A = A - B	**A = A + B**	**A = A / B**	**A = A * B**
C = A * 2	Correct	1st operator incorrect	1st operator incorrect	1st operator incorrect
C = A + 2	2nd operator incorrect	Both operators incorrect	Both operators incorrect	Both operators incorrect
C = A – 2	2nd operator incorrect	Both operators incorrect	Both operators incorrect	Both operators incorrect
C = A / 2	2nd operator incorrect	Both operators incorrect	Both operators incorrect	Both operators incorrect

Figure 1: Every possible operation for the program

Following every possible operation for the program above, we have listed out all every combination possible that would be incorrect, we can then design the test cases to validate between the correct and incorrect operation.

Task 2:

To determine whether the provided test case can achieve the required testing objective, we can use it as the input for our program. This can be further visually represented below:

(A =3, B = 1)	**A = A - B**	**A = A + B**	**A = A / B**	**A = A * B**
C = A * 2	4	8	6	6
C = A + 2	4	6	5	5
C = A – 2	0	2	1	1
C = A / 2	1	2	1.5	1.5

Figure 2: Every possible operation using given test case.

As we can see in the figure above, if using the given test case, namely A = 3 and B = 1, there will be a matching between our correct operation with an incorrect operation. This, as a result, will not be sufficient enough to achieve the required testing objective.

Task 3:

When taking a look at the provided test case in Task 2, we can see that if the result of A is 2, -2, 4, -4, our test cases would likely be failed as our result is affiliated with number 2. Furthermore, if the result of A is 0, it would also fail as we have multiplication operator. Therefore, we should choose test cases ensuring that the result of A would not be 2, -2, 4, -4 or 0. Our selected test cases would be.

Test case 1: A = 5, B = 2

(A = 5, B = 2)	A = A - B	A = A + B	A = A / B	A = A * B
C = A * 2	6	14	5	5
C = A + 2	5	9	4.5	4.5
C = A – 2	1	5	0.5	0.5
C = A / 2	1.5	3.5	1.25	1.25

Figure 3: Test case 1

Test case 1 is selected so that the result of A in any operation would not be 2, -2, 4, -4 or 0, by adhering to that rules we have created a concrete test case.

Test case 2: A = 9999, B = 10002

(A = 9999, B = 10002)	A = A - B	A = A + B	A = A / B	A = A * B
C = A * 2	-6	40002	1.9994001199760048	200019996
C = A + 2	-1	20003	2.9997000599880024	100010000
C = A – 2	-5	19999	-1.000299940011997	100009996
C = A / 2	-1.5	10000.5	0.4998500299940012	50004999.0

Figure 4: Test case 2

Test case 2 is selected so that the result of A in any operation would not be 2, -2, 4, -4 or 0, furthermore, A and B are large numbers, by adhering to that rules we have created a concrete test case.

Test case 3: A = -63, B = -64

(A = -63, B = -66)	A = A – B	A = A + B	A = A / B	A = A * B
C = A * 2	2	-254	1.96875	8064
C = A + 2	3	-125	2.984375	4034
C = A – 2	-1	-129	-1.015625	4030
C = A / 2	0.5	-63.5	0.4921875	2016.0

Figure 5: Test case 3

Test case 3 is selected so that the result of A in any operation would not be 2, -2, 4, -4 or 0, furthermore, A and B is negative numbers, by adhering to that rules we have created a concrete test case.

Task 4:

For this task, we will write a Python program to identify all invalid values for A to become the test case if B is 1, the program and a brief explanation will be included in **Appendix A**. The figure belows is the output for my program.

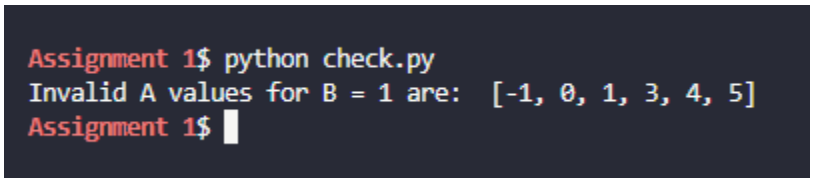

Figure 6: Program's output

We can see why these test cases fail from the analysis in Task 3 that the result of A in any operation should not be 2, -2, 4, -4, or 0. For cases of A is -1, 1 or 3, there would be operation that results in A equals 2 or -2. For cases of A are 3, 4 or 5, there would be operation that results in A equals 4. Finally, if A equals 0, there would be operation that results in A equals 0. All of these cases would result in an invalid test case for our program.

Appendix A

```
def correct_operations(A, B):
    A = A - B
    C = A * 2
    return C
def test_operations(A, B):
    incorrect_operations = [(A + B) + 2,
                            (A + B) - 2,
                            (A + B) * 2,
                            (A + B) / 2,
                            (A - B) + 2,
                            (A - B) - 2,
                            (A - B) / 2,
                            (A * B) + 2,
                            (A * B) - 2,
                            (A * B) * 2,
                            (A * B) / 2,
                            (A / B) + 2,
                            (A / B) - 2,
                            (A / B) * 2,
                            (A / B) / 2]

    for result in incorrect_operations:
        if result == correct_operations(A, B):
            return True
    return False
def main():
    invalid_testcase = []
    for A in range(-100, 101):
        if test_operations(A, 1):
            invalid_testcase.append(A)
    print("Invalid A values for B = 1 are: ", invalid_testcase)
if __name__ == "__main__":
    main()
```

Figure 7: Program to evaluate invalid value for A with B equals 1

The '**correct_operations**' function computes the correct result by subtracting *B* from *A* and doubling the result. The '**test_operations**' function checks if any of several incorrect operations match this correct result. The '**main**' function tests this for values of *A* from -100 to 100 with B = 1and collects all values of *A* for which any incorrect operation matches the correct result, printing these values.