```python
import unittest
from calculator import add, subtract, multiply, divide, modulus

class TestCalculator(unittest.TestCase):
    def test_divide_typo(self):     # 16. The following test method is added,
but something is wrong:  What is the problem with this test?
         divide(10, 2)

    def test_add(self):
        print(add(5, 5)) # 5. What is the output of add(5, 5)?
        self.assertEqual(add(10, 5), 15)
        self.assertEqual(add(-1, -1), -2)
        self.assertEqual(add(1, 1), 2)          # 19.


    def test_subtract(self):
        print(subtract(10, 15)) # 6. What is returned by subtract(10, 15)?
        self.assertTrue(isinstance(subtract(10, 2), int)) # 17. You are asked
to verify that the subtract() function never returns a string. Write a test to
check the return type of subtract(10, 2). What assertion would pass?
        self.assertEqual(subtract(10, 5), 5)
        self.assertEqual(subtract(0, 5), -5)

    def test_multiply(self):
        print(multiply(3, -3)) # 7. What does multiply(3, -3) return?
        self.assertEqual(multiply(3, 4), 12)
        self.assertEqual(multiply(3, 0), 0)

#         20.A developer accidentally changes this line:
            # def multiply(a, b):
            #         return a * b + 1
        self.assertEqual(multiply(3, 3), 9)
        self.assertEqual(multiply(0, 3), 0)
        self.assertEqual(multiply(5, 5), 25)

    def test_divide(self):

        print(divide(10, 2))  # 8.The function divide() always returns an
integer. (returns 5.0 (float), not 5 (int))
        self.assertEqual(divide(10, 2), 5.0)
        self.assertEqual(divide(5, 2), 2.5)

    def test_divide_by_zero(self):
        # divide(1, 0) # 10. What exception is raised when you run divide(1,
0)?
        self.assertEqual(divide(10, -2), -5.0) # 13. To test equivalence
partitioning in divide(), which test checks behavior when a negative divisor
is used?
```

```
        with self.assertRaises(ValueError): # 11. test_divide_by_zero()
verifies exception handling
            divide(10, 0)
            divide(5, 0) # 15. You're conducting white-box testing to ensure
the if b == 0 block in divide() is covered. Which test achieves this?

    def test_modulus(self):
        print(modulus(14, 4)) # 9. What is the correct output of modulus(14,
4)?
        print(modulus(11, 3))  # 14. All current modulus tests use even
numbers as divisors. What result would you get from modulus(11, 3)?
        self.assertEqual(modulus(5, 10), 5) #18. What is the actual output
when this test is run?
        self.assertEqual(modulus(10, 3), 1)
        self.assertEqual(modulus(10, 2), 0)

    def intentionally_fail(self):# 19.You’re told one of the tests is written
to intentionally fail. Which test below would definitely fail?
        self.assertEqual(add(1, 1), 2)
        self.assertEqual(divide(4, 2), 2)
        self.assertEqual(modulus(10, 3), 2)  #(answer)
        self.assertEqual(subtract(5, 2), 3)

if __name__ == '__main__':
    unittest.main()


# 12. You notice that test_add() only tests positive and negative values. You
want to test the identity property of addition. Which of the following test
cases best represents this?
# self.assertEqual(add(0, 99), 99)
```