For this deliverable, we injected 5 faults into the source code we were testing. After injecting these faults, 8/25 of our test cases passed. Before injecting these faults,  22/25 of our test cases passed.This demonstrates that our tests test a wide variety of situations. If so many of our tests hadn't failed after injecting faults, it would have shown that our test data was of poor quality. We attempted to inject faults that were bad enough but didn't completely annihilate the functionality of the code. This was a fun experience.

The faults we injected are as follows:

1. Modified the lcm method from
    a. return Math.abs(arg0 * arg1) / MathOps.gcd(arg0, arg1);
    b. to
    c. return Math.abs(arg0 * arg1) / MathOps.gcd(arg0, arg0);
    d. We expected this to cause all test cases with arguments that aren't equal to fail.
2. Changed gcd loop while condition from rest > 0 to rest >= 0.
    a. We expected this to cause an infinite loop and cause all test cases to fail. This is precisely what happened. Since lcm uses this method, we expected it to also cause all test cases for lcm to fail, which is what happened.
3. In argMax, changed entries[element] > entries[argmax] to entries[element] < ….
    a. We expected this to cause all test cases to fail except for 0, 50
4. In argMin, changed entries[element] < entries[argmax] to entries[element] > …
    a. We expected this to cause all test cases to fail except for 0, 50
5. Changed dbl1 - dbl2 to dbl1 + dbl2 in approxEqual
    a. We expected this to cause only 1, 1 to fail