1. Code:

```
// Uses Euclid's algorithm to calculate the GCD.
 provate long GCD( long a, long b )
 {
    a = Math.abs( a );
    b = Math.abs( b );

    for( ; ; )
    {
       long remainder = a % b;
       If( remainder == 0 ) return b;
       a = b;
       b = remainder;
    };
 }
```

2. Either writing the code to explain what the code does (rather than what it should do), or writing code that is just English versions of the code that follows.
3. Have the code fail with numbers that have no GCD. The current version hides this error.
4. The code would send an argument exception or similar if the arguments have no GCD.
5. Drive car into parking lot, find empty space, drive car into empty space, put car in park. This assumes there is a car and it is functional, and that the supermarket has a parking lot.
6. .

```
function testIsRelativelyPrime() {
    const testCases = [
        { a: 21, b: 35, expected: false },  // 21 and 35 are not
relatively prime
        { a: 3, b: 7, expected: true },      // 3 and 7 are relatively
prime
        { a: 0, b: 5, expected: false },     // 0 and 5 are not
relatively prime
        { a: -1, b: 100, expected: true },   // -1 and 100 are
relatively prime
        { a: 0, b: 0, expected: false },     // 0 and 0 are not
relatively prime
        { a: 1, b: -1, expected: true }      // 1 and -1 are
relatively prime
    ];

    testCases.forEach((testCase, index) => {
        const { a, b, expected } = testCase;
        const result = IsRelativelyPrime(a, b);
```

```
        console.log(`Test Case ${index + 1}: IsRelativelyPrime(${a},
${b}) => ${result === expected ? 'Pass' : 'Fail'}`);
    });
}
```

7. I used black-box testing and tested some inputs with their expected results.
8. My testing code worked, as it was just used to find the expected results.The testing code ensured that the function worked as expected.
9. It is black-box, as you do not tailor your tests to your knowledge of how the code functions.
10. There is estimated to be 50 bugs total, with 40 still at large.
11. It means that there are an indeterminate number of bugs estimated via the Lincoln estimate. Because the Lincoln estimate is already generally an underestimate, you cannot get a lower bound estimate.