```
// Use Euclid's algorithm to calculate the GCD.
// Declare method that takes in two longs to calculate gcd on
         provate long GCD( long a, long b )
         {
            // Get the absolute value of a and b, negative values won’t work
            a = Math.abs( a );
            b = Math.abs( b );


            //Repeat until we're done, for loop iterate until base case is
met
            for( ; ; )
            {
               // Set remainder to the remainder of a / b, declaring our base
case
               long remainder = a % b;
               // If remainder is 0, we're done.  Return b.
               If( remainder == 0 ) return b;
               // Set a = b and b = remainder. Reset variables for next
iteration of loop
               a = b;
               b = remainder;
            };
         }
```

2. It could be that the programmer wrote these comments after completing all the code. Comments should concisely describe exactly what a method does.

3. This program does a good job of error checking. It validates both the input and output, and it has a debug.assert statement.

4. No, error handling should be done by its caller.

5. Driving to supermarket
   - Open door to leave the apartment
   - Walk to the elevator after taking a right out the front door
   - Go down to P1 in the elevator
   - Return to previous parking spot
   - Unlock the car and get in the driver’s seat
   - Put your seatbelt on and start the car
   - Reverse out of spot and exit garage
   - Take right at garage exit

- Take right at apartment exit
- Take right on Lincoln Blvd
- Take right into Ralphs parking lot
- Assumptions: driving from PDO to Ralphs, person can drive

6.
```
are_rel_prime_test(a,b) {
      a = Math.abs(a);
      b = Math.abs(b);

      if( a=1 || b=1 ) return true;
      if( a=0 || b=0 ) return false;

      min = min(a,b)
      for (factor=2; factor <= min; factor++){
            if((a%factor == 0) && (b%factor == 0)) return false;
      }
      return true;
```

7. Must use black box because we don't know how the method is written. If we had code for the method, we could use grey and/or white box testing.

8.

9. Black box testing must be exhaustive because you don't have any knowledge of the inner workings of the method.

10. We get a lower bound of 14 after averaging between the three testers and we get an upper bound of 20.

11. It's not possible to find a lower bound with no bugs in common because we would be dividing by zero. Make the lowest common error >=1 to avoid this.