**CMSI 4072 HW3**



// Use Euclid's algorithm to calculate the GCD.

private long GCD( long a, long b )

{

a = Math.abs( a );

b = Math.abs( b );

// Finds the remainder between a and b, until a value leaves a

remainder of 0

for( ; ; )

{

long remainder = a % b;

If( remainder == 0 ) return b;

a = b;

b = remainder;

};

}

1. 1. Writing comments that explain what the code currently does, rather than what it should do
   2. Writing comments as you write the code
2. Offensive programming can be used to enforce the inputs into the function. For example, the number must be positive, and it must fit within some reasonable range for example [0, 999999].

if ((-999999 > a || a > 999999) || (-999999 > b || b > 999999)) {

throw new IllegalArgumentException("Arguments must be between -999,999

and 999,999");

}

1. Yes, error handling should be used to tell the user they’ve done something wrong – not just send out some error that is difficult to understand.
2. 1. startCar()
      1. engineOn()
   2. followRoute()
      1. If (turn == left) { turnLeft()}
      2. If (turn == right) { turnRight()}
      3. Else {straight()}
   3. followTrafficLaws()
      1. If (light==red){stop()}
      2. If(light==green){go()}

private boolean isRelativePrimeTest(long a, long b){

if (GCD(a, b) == 1 || GCD(a, b) == -1) {

if (isRelativelyPrime(a, b) == true){

return true

} else {

return false

}

} else {

if (isRelativelyPrime(a, b) == false){

return true

} else {

return false

}

}

}

1. White-box testing is used above, since I know how the method should work, I can test specific cases knowing what the outcome should be. Black-box could work, as by using random inputs we can test how the function handles the test cases – however, some knowledge of how the method is expected to work is needed as we still need to evaluate the output of the test cases. Additionally, exhaustive would be feesible as with a range of -1000000 to 1000000 of input values provides a set number of values to iterate over and see if anything unexpected happens.
2. I found no bugs, but testing is still a necessity to make sure things work as expected.
3. Exhaustive testing is white-box testing, as there needs to a known feasible amount of cases to test.
4. The Lincoln index can be used to make a formal guess as to how many bugs remain by doing the calculation for all possible combination of testers.
5. The Lincoln index would return undefined if two testers find no bugs in common, this mean that there is no accurate guess to be made in this case. A lower bound estimate could just be a sum of the bugs found by each tester.