Nicholas Soffa

Professor Johnson

CMSI 402

13 March 2017

Assignment #3

7.1 // Use Euclid’s algorithm to calculate the GCD.

// Divide the bigger number, a, by the smaller number, b, and unless the remainder is 0 replace a with b, b with the remainder and repeat until the remainder is 0

// See en.wikipedia.org/wiki/Euclidean\_algorithm for more info

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;

};

}

7.2 The programmer might have written the comments right after writing code and used top-down design, but it went through so many bug fixes that it wouldn’t match, or the programmer wrote comments at the end of writing code and wrote JBGE or just barely good enough comments

7.4 The validation code for 7.3 is already offensive programming with validating the input and result, and the Debug.Assert takes care of any problems.

7.5 No you don’t because you want the calling code to handle errors.

7.7 1. Acquire Keys.

2. If CarLocked() == true

a. Push Unlock button while CloseToCar() == true.

b. if CloseToCar() == false, WalkToCar() until CloseToCar() == true;

3. Sit In Driver’s Seat

4. Start Engine with keys

5. If OpenPathToStreetForward() == true

a. Put car in drive

b. Push Gas pedal to get on to street

6. if OpenPathToStreetForward() == false

a. Put car in Reverse

b. Push Gas pedal to get on to street

7. Assuming driver knows route to supermarket and doesn’t need a map or gps, and car is in perfect condition and isn’t about to break down, steer car down correct streets until InSuperMarketParkingLot() == true

a. If (mph > speedLimit) {SlowDown()}

b. If (CarInFront) {SlowDown()}

c. If (RedLight || Yellow Light || StopSign) {SlowDown()}

d. If (Construction) {SlowDown()}

e. If (PulledOver) {SlowDown(); PullOver();}

8.1 private bool Test\_IsRelativelyPrime ( int a, int b) {

Make sure a is positive by taking absolute value

Make sure b is positive by taking absolute value

check if either a or b is one, then return true

check if either a or b is zero, then return false

Find which is smaller, a or b

for loop from 2 to minimum of a and b

if a or b modulo the iterator == 0 then return false

return true if no false’s are returned

Then test a bunch of different cases with random numbers between -1,000,000 and 1,000,000 like a & b, a & a, a & 1, a & -1, a & 0, a & -1,000,000, a & 1,000,000, -1,000,000 & 1,000,000, and vice versa for all of those cases.

8.3 Black-Box since we don’t know how AreRelativelyPrime works. If we were told how it works, then we could use white or gray box tests. Exhaustive test could be possible if the extreme values weren’t so large.

8.9 They are black box tests because they don’t rely on what’s going on inside the method they are testing

8.11 You could make 3 pairs to calculate the Lincoln Indexes. Alice & Bob got 10, Alice & Carmen got 12.5, and Bob & Carmen got 20. The average of all 3 is 14 bugs. Plan for the worst which is Bob & Carmen so 20 but keep track of bugs found to change the estimate later on.

8.12 If the testers don’t find any bugs in common then the Lincoln estimate divides by 0 and you get infinity. Which means you have no estimate to go off of. You can get a lower bound by pretending that they had 1 bug in common.