Justin Sanny

3/12/17

Assignment 3

1. (7.1) Explain the algorithm and how it works instead of obvious comments. You can link the URL of Euclid’s algorithm in the comments instead.

2. (7.2) The programmer added comments after writing the code and only explaining what the code does and not why it does it. Another way is that the programmer goes into excruciating details instead of simply giving a url to explain the entirety of the code.

3. (7.4) We can use offensive programming by offensively checking the variables and throwing an exception if there is a problem.

4. (7.5) No, we don’t have to add error handling code.

5. (7.7) 1)Find Car

2) Unlock Car

3) Open Car

4) Start Car

5) Make a U-Turn

6) Go down until intersection.

7) Turn left until street ends.

8) Turn right, keep going until intersection.

9) Turn left

10) Go till you hit the super market

11) Park car in empty space

12) Go buy stuff.

Assumption: Car is not in the garage.

You didn’t die on the way there or get in an accident.

6. (8.1)

public bool Validate\_AreRelativelyPrime(int a, int b){

a = Math.Abs(a);

b = Math.Abs(b);

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

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

int min = Math.Min(a, b);

for (int x = 2; x <= min; x++){

if ((a % x == 0) && (b % x == 0)) return false; }

return true;

}

7. (8.3) You would have to use black-box test since we don’t know how the code works.If we knew how it did work then we could go a white-box or gray-box. Exhaustive is an option but there would be too many values to go through.

8. (8.5) After implementing, I found that I lacked restrictions for the numbers. Using tests, helped me figure out special cases and overall cases that should’ve been in the ode in the first place.

9. (8.9) Exhaustive testing goes into black-box testing because neither relies on knowledge about the program.

10. (8.11) Have the people calculate three different Lincoln indexes: Alice/Bob, Bob/Carmen, Carmen/Alice. Then you can find the average which is 14 bugs or assume the worst case and find that there’s 20 bugs.

11. If there are no bugs in common then you have to divide by 0 which is impossible. To get a lower bound, assume that there is at least 1 bug in common which will help assume the number of bugs.