**Coverage Part 3 – Input Domain Analysis**

**2(a)**

The important thing to recognise about input domain analysis is that, rules about disjoint and complete partitions aside, it is a subjective process, the success of which is often dictated by the ingenuity of the tester.

While the lecture already provides a “good” solution, a tester may seek to adjust the characteristics/partitions to reduce the final number of test cases, or to take account of any further corner cases they have identified.

A further characteristic, producing a new set of partitions, could be the relation of the side lengths to one another, for example side1 + side2 >= side3 – true, false; etc. This would add another dimension, from an interface-based perspective, regarding the validity/invalidity of a triangle. Or, the initial side lengths with respect to the order they’re provided to the method to check whether the method correctly sorts them.

**(b)**

Combining a negative length with an equilateral triangle is one example of a “cannot be combined constraint”. Inversely, a positive length with a valid triangle type is a “must be combined”.

**(c)**

Again, no one particular right answer here. Another sensible strategy for avoiding invalid combinations would be to choose valid/invalid partitions for a characteristic as a base choice when combining with other valid/invalid partitions.

**(d)**

There is a bug in the program! See if you can find it.