Study questions for Lecture 3 - Answers

a) Consider the multi/exp example. Suggest 2 ways to fix the interface problem

In this example mult() has two inputs, x and y. It works by adding up x y times. Hence y has to be non-negative. exp() has two inputs, v and w. It works by multiplying v times itself w times. w has to be non-negative, but v can be negative. exp() calls on mult() to do its multiplying. Since v can be negative, there will be an interface problem if the call to mult() is of the form mult(c,v) where c is the partially computed exponent, because v could be negative. Inside exp() c is always positive.

The way to avoid/fix the interface problem is to ensure that mult(y,z) is always called with v as the first term, not the second. Another approach is to restrict exp() to having both arguments non-negative. Another approach is to re-write mult() so that both arguments can be negative. This will require setting the counter to the absolute value of the input y.

b) For a repaired version of the exp()/mult() example, construct an assertion oriented verification of exp() similar to the one carried out for mult()

exp(v,w):
   input v, w  A1: {v, w integer, w >= 0}
   exp = 1;
   count = w;
   label: A 2: { v**w = exp*(v**count), exp and v are integer}
   if count == 0 then
      return exp;
   A3: {exp = v**w}
   count = count -1
   A4: {v**w = exp*(v**count+1), exp and v are integers}
   exp = mult(exp,v)
   A5: { v**w = exp*(v**count), exp is integer}
   go to label

A1 -> A2: exp*(v**count) = v**w, so v**w = exp*(v**count) is satisfied. If v is an integer it still is an integer because it was not changed. exp is an integer because it is = 1.

A2 -> A4: Since exp and v are not changed on this subpath, if they were integers before they still are. Also, since count was decremented, the previous relationship implies that the new one holds, with count incremented so compensate for its reduced value. Since exp and v are integers, we satisfy the preconditions for the use of mult().

A4 -> A5: exp will be an integer here, because mult() is taking the product of two integers, which will be an integer. The relationship holds because A4 held
previously and we have reduced the value of the exponent (count+1) to count to compensate for the increase in value of exp.

A5 -> A2: A2 holds when coming from A5 because the relationship is the same, exp is integer, and v has not been changed. So if v was integer before, it still is now.

A2 -> A3: if we take the branch out to A3 then we know count ==0. Hence v**count = 1, so that we have v**w =exp*(v**count) = exp.