Assignment 4 hints and abridgements, second portion

I am finding my sample solution for Assignment 4 reasonably lengthy. I offer the following hints in the hope that they may help, and I also offer some abridgements to allow you to omit (without penalty) some possibly tedious portions of the solution.

QUESTION 2: I found it useful to prove (and use) the following claim, directly from Definition 6.6. If F is a first-order formula that does not include the variable x, then $\forall x F$ is equivalent to F. Using double-negation, this means that $\exists x F$ is equivalent to F.

QUESTION 4: The three recursive calls to `groupTweak` permute the parameters `from`, `to`, and `intermediate`. This might make it confusing to relate parameter values to the parameter names in the preconditions and postconditions. One way around this is to assume WLOG that at the top level `from` is 0, `to` is 1, and `intermediate` is 2 (the particular choice of values shouldn't affect the argument, so long as they are distinct). Then, as the parameters permute, you can trace out which value is associated with which parameter.

This algorithm has more than a passing resemblance to the Tower of Hanoi (TOH) puzzle. If you are familiar with the puzzle, then imagine that the elements of `digit` are the rings (smaller rings have lower indices), and value of each element indicates which post it is on. The method `tweak` is meant to resemble the TOH rule restricting ring movements. Starting with all elements of `digit` with the same value corresponds to starting the TOH with all the rings on the same post. Does the model work?

I also have an iterative version, which I decided not to include in the assignment.

QUESTION 5: The algorithm contains three `for` loops, as well as the main `while` loop. You may assume, without proof, that the `for` loops terminate (otherwise, proof of termination is very easy). You may also assume without proof that the `for` loop that initializes `digit` initializes it to *base* empty Vectors, and that the `for` loop that calls `numList.addAll(digit[i])` makes `numList` equal to the concatenation of `digit[0]` $\cdots$ `digit[base-1]`.

You will still need to determine (and prove) what the other `for` loop (that copies elements from `numList` into the various `digit[i]`) does. This requires developing a precondition and postcondition, and proving that the loop is correct with respect to them. Think about how the values are ordered mod *magnitude* and mod (*magnitude* $\times$ *base*).

QUESTION 6: You may assume without proof that the greatest common divisor of (a, b) exists for natural numbers a and b. Implicit in the algorithm is that the greatest common divisor of (a, b) equals the greatest common divisor of $(b, a\%b)$. This (possibly useful) fact can be proved without induction.