Hints for Homework 8

You don't have to rewrite your functions if you don't use these hints, unless they indicate that you've misunderstood the problem.

Please program in an applicative style, using lists. That is, for this homework you may *not* use the imperative constructs of LISP (that is, `set`, `setq`, `setf`, `rplaca`, `rplacd`, `nconc`, or `delete`), arrays, or the "program feature" (`prog`, `prog1`, `prog2`, or `progn`). You may assume that argument lists are not circular.

1. (a) First define a function `nth-element`, such that `(nth-element 7 l)` returns the seventh element of `l`.
   (b) Try programming the problem description.
   (c) Try programming the problem description.
   (d) Use Euclid's algorithm, which takes advantage of the fact that `(gcd m n)` is equal to `(gcd n (rem m n))` if `n` is nonzero.
   (e) Try using the built-in XLISP function `member`. Don't name the second formal argument `t`.
2. You probably need to use `append` to define `postorder`. Note that `cons` only takes two arguments. Don't name your formal argument `t`.
3. This problem seems clear.
4. If you test this, say `(filter minusp '(0 -1 2))` or some such. Don't quote the first argument to filter, it should expect a function, not an atom as an argument.
5. Reduce takes three (3) arguments!
6. Use `mapcar2`, which you can define recursively. But don't use any other recursion.
7. In the body of the function you are to define only the following names should appear: `reduce`, `append`, `cons`, `nil lambda`, and the names of any formals for any lambdas you use.
8. Define `map` and `dup` first.
9. Matricies are represented as lists of lists. Use the functional `map2`.
10. Use an auxilliary function and lambda expressions.