Homework 3

This homework is due Friday, Feb. 16 (in class).

The Scheme program for problem 3 should also be written in a functional style, as for homework 2. Mail the code to `coleman` on snowhite.

1. (10 points) Using the encoding given in the lambda calculus handout in the course reference pack for the natural numbers (i.e., the Church numerals), give lambda terms that implement:
   (a) multiplication, and
   (b) exponentiation.
2. (10 points) Formally describe a terminal transition system for context-free grammars.
3. (30 points) For this exercise, you will implement a simple form of terminal transition systems in Scheme. The simplifications are that your program will not have to handle pattern matching or inference rules.

   To be specific you are to implement a function `tts` that takes as an argument a list of transition rules of the form `(==>` E_1 E_2`)`, where E_1 and E_2 are S-expressions. That is, S-expressions are used for input (and output) of configurations. The result of an application of `tts` should be a function that, when applied to an S-expression (encoding a configuration) returns a S-expression encoding a final configuration, if there is one. A final configuration is a configuration to which no rules apply. Use `equal?` to test whether a transition rule applies to a given configuration.

   For example, the result of the following expression should be `Z`.

   ```
   (let ((my-system (tts '((==> x y) (==> y z)))))
     (my-system 'x))
   ```

   Note that transition systems need not be deterministic, nor must they always be terminating.