CSE 341, Autumn 2008, Assignment 3
Scheme Warmup
Due: Monday October 20, 10:00pm

16 points total (2 points each for Questions 1–3; 10 points for Question 4)

You can use up to 2 late days for this assignment.

1. Write a function `cone_volume` that takes two numbers representing the height and radius of the base of a cone, and that returns the volume of the cone.

2. Write a function `take` that takes an integer `n` and a list `s`, and returns a new list consisting of the first `n` elements in `s`. Handle the cases of the length of `s` being less than `n`, and `n` being 0 or negative, in the same way that Haskell does for its definition of `take`.

3. Write a function `cubes` that takes a list of integers and returns a list of the cubes of those integers. For example, `(cubes '(1 3 10))` should evaluate to `(1 27 1000)`. `cubes` should use the built-in `map` function in Scheme. Don't define a named helper function – use an anonymous function.

4. This question uses materials in Chapter 2 of the book *Structure and Interpretation of Computer Programs*. The full text is available online (linked from the Scheme page); there is also a copy on reserve in the Engineering Library. Extend the symbolic differentiation program in Section 2.3.2 of SICP, as follows. First, load the symbolic differentiation program into Scheme and try it, to make sure it is working OK and that you understand it. (A copy is linked from the Scheme 341 page.) Now add the following extensions, by allowing the expressions to be differentiated to include:

   - the difference operator
   - sin and cos
   - raising an expression to an integer power

   Difference should be really easy — use sum as a model. The rules for the others are as follows:

$$\frac{d(\sin u)}{dx} = \cos u(\frac{du}{dx})$$

$$\frac{d(\cos u)}{dx} = -\sin u(\frac{du}{dx})$$

$$\frac{d(u^n)}{dx} = nu^{n-1}(\frac{du}{dx})$$

   For simplification build in the rules that anything to the 0 power is 1, and anything to the power 1 is itself. Your code for this question should be written entirely in a functional style — no side effects.

   Demonstrate your code works correctly on a well-chosen range of examples.

Turnin: Turn in your Scheme program and a script showing it running on some well-chosen test cases. As usual, your program should be tastefully commented (i.e. put in a comment before each function definition saying what it does).