Comp 210
Midterm Exam
1997 Spring

Instructions:

This exam is a three hour, take-home exam. The exam is closed book, closed notes. You may NOT use a computer on this exam. However, you may prepare and use one 8.5in x 11in crib sheet. Each student must prepare his or her own crib sheet. When done, please attach the crib sheet to your exam. You do not need to provide program templates unless otherwise indicated.

1. [20 points] Write a program that models an Ironman in the Beer-Bike competition as an object (without using global variables). An Ironman object is a function that accepts a symbol (representing an action) as its only argument. The valid actions are *'ride*, *'chug*, and *'intox-level*.

   The *'chug* action increments by one the number of beers currently ingested. (Initially, no beers have been ingested.)

   When invoked on the argument *'intox-level*, the Ironman function returns the number of beers currently ingested.

   If an Ironman has currently ingested more than three beers, then the *'ride* action causes the Ironman to puke, with the result that the number of beers ingested is reset to zero. Otherwise, the ride has no effect on the Ironman's internal state.

2. [15 points]
The *composition* of two single-argument functions $f(x), g(x)$ is the function $f(g(x))$.

The expression $f^{(n)}(x)$ denotes the function f composed with itself n times. The expression $f^{(0)}$ denotes the identity function; that is $f^{(0)}(x) = x$. For example, $f^{(3)}(x) = f(f(f(x)))$.

Your overall task will be to write a function *Nth*, which takes a single-argument function f and a natural number n, and returns the function $f^{(n)}$.

(a) [5 points] Write a function *compose* which takes two single-argument functions f, g and returns a single-argument function which is the composition of f and g.

(b) [5 points] Give the template for functions which take a natural number and a function as input, and which recur on the natural number.

(c) [5 points] From the template, write the function *Nth*, using the function *compose* as a helper function.

3. [15 points]

   (a) [5 points] Write a Scheme function *sumsq-rec* which computes the sum of the squares of a list of numbers, using natural recursion.

   (b) [5 points] Write a Scheme function *sumsq-acc* which computes the sum of the squares of a list of numbers, using accumulator-style recursion.

   (c) [5 points] If inexact (floating-point) arithmetic is used, do these two functions return the same value for all possible inputs? If not, show a list of numbers for which these functions return different results. (As in lab, presume floating-point arithmetic uses 15 significant decimal digits of accuracy.)