COMP 210, Fall 2000, Homework 7
Due Monday, October 30, 2000 at the start of class

Before you start the homework, you should remind yourself of our General Advice, Advice on Homeworks, and Grading Guidelines. All are available from the class web site (http://www.owlnet.rice.edu/~comp210).

In deference to the exam that you have this week, this homework covers about 1/2 of the material in a normal homework. It is worth 1/2 of the points of a normal homework. I recommend that you do the homework before the exam.

---

1. (2 pts) DrScheme has lost its mind. Somehow, it has lost the ability to compare natural numbers. All of <, <=, =, >=, and > have been lost. Your mission is to save the day by implementing a function **greater? : natnum natnum -> boolean** that returns true if its first argument is larger than its second. Both arguments are assumed to be natural numbers in the sense defined in this class. (See notes for lecture 4, available on the web site.) To complicate matters, the only numeric functions you can use are **sub1** and **zero?** . (You can use any of the boolean functions, such as **and**, **or**b **not**, **…**)

   a) Hand in the case analysis that you do to generate the template. (See lecture 16.) Show your template as well as the final code.

   b) Develop a version of **greater?** that operates on general numbers–-including non-integers and non-positive numbers. You may need additional functions to implement this version, including **abs**, **positive?**, and **negative?**. Hide any helper functions that you develop inside a **local**.

2. (1 pt) Given the following definition for a point

```
;; a point is
;;    (make-point x y)
;; where x and y are numbers
(define-struct point (x y)
```

   Use the Scheme functions **filter**, **length**, and **sqrt** to create a function **within-1** that consumes a list of point and produces a list of point. The list that it produces should contain exactly those points that are within a distance of 1 from the point (0,0).

   [That is, the square root of $(x-0)^2 + (y-0)^2$ is less than or equal to one.]

   Use **local** to hide any helper functions that you write.

3. (2 pts) You are to build a simple model of how the Social Security Administration (SSA) might project the cost of benefits. Assume that each taxpayer has a record

```
;; a taxpayer is a
;;    (make-taxpayer  name age)
;; where name is a symbol and age is a number
(define-struct taxpayer (name age))
```

Of course, the SSA keeps a list of all the taxpayers who are involved in the Social Security System. Once a year, they need to increment the age of each taxpayer and count the number of taxpayers old enough to receive benefits.

Write two functions:

a. Write **eligible: list-of-taxpayer -> list-of-symbol**. Your **eligible** function should return a list of the names of each taxpayer who is over 65.

b. Write **make-older: list-of-taxpayer -> list-of-taxpayer**. Your **make-older** consumes a list of taxpayers and returns a list in which each taxpayers age has been increased by one. The SSA runs this at the beginning of each year.

You should use Scheme's abstract functions (**filter**, **map**, **foldr**, and **foldl**) where possible in your implementation. Use lambda for any helper functions that you pass into the abstract functions.