CSci 4651 Spring 2006
Problem Set 2: Functional programming (Scheme). Due Wedn.
Feb. 8th at 8pm.

Problem 0. If you haven’t finished the lab, please finish the remaining lab
problems and submit them as a part of this problem set.

Problem 1. Scheme allows a programmer to write very general functions
which can be instantiated to perform a variety of tasks. Below is a function
traverse that allows working with lists in a very general way. traverse returns
a function that traverses a list and performs a specified task. The task depends
on the parameters passed to traverse.

Given appropriate parameters, traverse can generate a mapping function
(a function that modifies all elements of a list in a certain way), a filter (creating
a new list that contains only the elements of the given list that satisfy a certain
condition), and functions for many other tasks on lists. Below is definition of
traverse:

(define traverse (lambda (combine do seed)
  (lambda (x)
    (cond ((eq? x '()) seed)
          (#t (combine (do (car x))
                      ((traverse combine do seed) (cdr x)))))))

The three parameters of traverse are as follows:

• combine is a function that combines the result for one element with the
  result for the rest of the list,

• do is a function that performs the specified action on an element, and

• seed is the result for an empty list.

Example: the function mapsquare below is defined via traverse. Given a
list of integers, it creates a list of squares of these integers:

> (define mapsquare (traverse cons (lambda (x) (* x x)) '()))
> (mapsquare '(1 -2 3))
(1 4 9)

Question 1. Using traverse, define and test the following functions:

1. sumlist to compute the sum of all the elements of an integer list.

2. count to count the number of elements in a list (make sure to test this
   function on a list of non-integers).

3. remove5 to remove all 5s from a list of integers.

4. reverse to reverse a list.
5. \texttt{min} to find a minimum element in a list of integers (What would be the seed for this function? Make an assumption about the largest number that may appear on a list)

6. \textbf{Extra Credit.} \texttt{bettermin} – a function that finds a minimum element in a non-empty the list and is undefined on an empty list (hint: if a variable is undefined, any predicate on this variable, such as \texttt{number?} or \texttt{list?} returns \#f). Your function should work for arbitrary large numbers.

You may define other functions to solve the problem. Submit a printout of your definitions (including all the tests) AND the result of their run.

\textbf{Question 2.} Write a function \texttt{deeptraverse} which is analogous to \texttt{traverse}, but works on lists of lists (of arbitrary level of nesting). For instance, you should be able to use \texttt{deeptraverse} like this:

\begin{verbatim}
> (define deepmapsquare (deeptraverse cons (lambda (x) (* x x)) '()))
> (deepmapsquare '(1 () (3 (-2 5))))
(1 () (9 (4 25)))
\end{verbatim}

The function \texttt{list?} which returns \#t if the argument is a list and \#f otherwise might be helpful for this task.

Test your solution carefully to make sure that it works for various kinds of nested lists.

\textbf{Question 3.} Using \texttt{deeptraverse} from Question 2, define the following functions:

1. \texttt{deepsumlist} to compute the sum of all the elements of a list of lists.

2. \texttt{deepreverse} to reverse every list in a list of lists. For example,

\begin{verbatim}
> (deepreverse '(1 () (3 (-2 5))))
(((5 -2) 3) () 1)
\end{verbatim}

3. \texttt{flatten} to “flatten” a list of lists, i.e. to put all of its elements in a single list. For instance:

\begin{verbatim}
> (flatten '(1 () (3 (-2 5))))
(1 3 -2 5)
\end{verbatim}

Note: your function should preserve the order of the elements.

\textbf{Problem 2.} Exercise 3.1 p. 40.
\textbf{Problem 3.} Exercise 3.2 p. 40-41, parts a,b,c.
\textbf{Problem 4.} Exercise 3.6 p. 44-45.