COMP 210, Spring 2001, Homework 8
Due Wednesday, March 21, 2001 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).

This assignment is abbreviated (5 Pts instead of 10 Pts) to give you time to study for the exam next Wednesday evening.

1. (2 Pts) Use the abstracted functions (functionals) **`map`**, **`foldl`** , and **`foldr`** to solve the following two programming problems. The solutions do not involve any explicit use of recursion.

   a. (1 Pt) Define a function **`max-list`** to compute the maximum of a non-empty list of numbers. Hint: recall the trick of using **`#i-inf.0`** as the maximum of the empty list to simplify the recursive decomposition of the problem.

   b. (1 Pt) Define a function **`scalar-prod`** that computes the scalar product of two lists of numbers of equal length. The scalar product of (*a1 a2 … an*) and (*b1 b2 … bn*) for $n \geq 0$ is defined as

   $$a1*b1 + a2*b2 + \ldots + an*bn$$

   Hint: the **`map`** function accepts binary functions as well as unary functions. Hence,

   ```
   (map f (list a1 a2 … an) (list b1 … bn)) =
       (list (f a1 b1) (f a2 b2) … (f an bn))
   ```

2. (3 Pts) Exercises on accumulators.

   a. (1 Pt) Recall the function **`split`** (which is not accepted as a function name in some versions of DrScheme because it is a keyword) used as a helper in the **`mergesort`** program in the last homework assignment. Write a definition for split that uses accumluators to form the "left" and "right" parts of the input list. Call your function **`split-list`** if your version of DrScheme does not accept the name **`split`**.

   b. Recall the function **`flatten`** from Homework 4 that flattens a **`list-of-list-or-symbol`** to a **`(list-of symbol).`** Write a definition for **`flatten`** that uses an accumulator to form the flattened list.

   c. Recall the function **`dup-names`** from Homework 5 that finds the duplicated names in a parent-based family tree. Write a definition for **`dup-names`** that uses an accumulator to record the names that been scanned so far.

3. (Extra Credit: 5 Pts) Define a Scheme data representation for boolean expressions constructed from variables, the unary operator **`not`**, and the binary operators **`and`**, **`or`**, and **`implies`**. Define a Scheme data representation for an *environment* that

maps some finite collection of variable names to boolean values (**`true`** and **`false`**). Define a Scheme function **`eval`** with the contract and header

```
eval: boolean-expression environment -> boolean
(define (eval be env) …)
```

that evaluates the boolean expression **`be`** in the environment **`env`**.