CS 257: Non-Imperative Programming: Scheme!
Homework 7 (Spring '06)

1. Exercises 7.12, 7.13, 7.18, 7.22, 7.26, 7.30, 7.31

2. The higher-order function, *tail-recur*, takes the following arguments:

   - *bpred* - a procedure of *x* which returns #t if the terminating condition is satisfied and #f otherwise.
   - *xproc* - a procedure of *x* which updates *x*.
   - *aproc* - a procedure of *x* and *acc* which updates *acc*.
   - *acc0* - an initial value for *acc*.

   and returns a tail recursive function of *x*. For example, it can be used to write the function, *factorial* as follows:

   ```
   (define factorial
     (tail-recur zero? (lambda (x) (- x 1)) * 1))
   ```

   Write *tail-recur*.

3. Use *tail-recur* to write *reverse*.

4. Use *tail-recur* to write *iota*.

5. The function *any?* takes a predicate, *pred*, as its first argument and applies it to the elements of its second argument, a list, *ls*. If any elements of *ls* satisfy the predicate, *any?*, returns #t otherwise *any?* returns #f. Use *tail-recur* to write a function, *any?-c*, which takes a predicate, *pred*, as its argument and returns a function of a list, *ls*. Use *any?-c* to define *any?*.

6. Define a function *clock-maker* which creates instances of a class, *clock*, representing a 12 hour clock, using three *restricted-counter* objects (See Exercise 12.4 in Springer and Friedman) to represent hours, minutes, and seconds. Clock instances should recognize the following methods:

   - *type* - Returns *'clock*.
   - *tic!* - Advances the time by one second.
   - *seconds!* - Set the second hand to the value of the first optional argument. Displays an error message if the argument is less than 0 or greater than 59.
   - *minutes!*- Set the minute hand to the value of the first optional argument. Displays an error message if the argument is less than 0 or greater than 59.
   - *hours!* - Set the hour hand to the value of the first optional argument. Displays an error message if the argument is less than 0 or greater than 11.

- *display* - Displays the current time in a HH:MM:SS format.

You can test your clock class using the following test routine:

```
(define clock-tester
  (lambda ()
    (let ((clock (clock-maker)))
      (letrec
        ((loop
          (lambda (seconds)
            (if (< seconds 3601)
                (begin
                  (send clock 'tic!)
                  (loop (add1 seconds)))))))
        (send clock 'hours! 11)
        (send clock 'minutes! 3)
        (send clock 'seconds! 47)
        (loop 0)
        (send clock 'display)))))
```

If your clock is working correctly, it should display 00:03:48.