CS 565 Out: February 20, 1997
Jens Palsberg Due: February 27, 1997

Homework 5:

Converting to Tail Form

Transform these procedures to CPS and then test them using the procedure `final-valcont` in *EOPL* p.279. You can assume that `atom?` is a primitive operation.

1.

```
(define list-of-symbols?
  (lambda (l)
    (if (null? l)
        #t
        (if (symbol? (car l))
            (list-of-symbols? (cdr l))
            #f))))
```

2.

```
(define remove-all
  (lambda (i l)
    (cond
      ((null? l) '())
      ((atom? (car l))
       (let ((rember-cdr (remove-all i (cdr l))))
         (if (eqv? i (car l))
             rember-cdr
             (cons (car l) rember-cdr))))
      (else (cons (remove-all i (car l))
                  (remove-all i (cdr l)))))))
```

3.

```
(define remove
  (lambda (i l)
    (letrec
      ((loop
         ;; (loop l) = (remove i l)
         (lambda (l)
           (if (null? l) '()
             (if (eq? i (car l))
               (loop (cdr l))
               (cons (car l)
                     (loop (cdr l))))))))
      (loop l))))
```

4.

```
(define reverse!
  (letrec ((loop
             (lambda (last ls)
               (let ((next (cdr ls)))
                 (set-cdr! ls last)
                 (if (null? next)
                     ls
                     (loop ls next))))))
    (lambda (ls)
      (if (null? ls)
          ls
          (loop '() ls)))))
```

5.

```
(define list-index
   (lambda (s los)
      (if (null? los)
          -1
          (if (eq? (car los) s)
              0
              (if (= (list-index s (cdr los)) -1)
                  -1
                  (+ 1 (list-index s (cdr los))))))))
```

6.

```
(define swapper
   (lambda (s1 s2 slst)
   (cond
      ((null? slst) '())
      ((atom? (car slst)) (cond
              ((eqv? s1 (car slst))
                      (cons s2 (swapper s1 s2 (cdr slst))))
              ((eqv? s2 (car slst))
                      (cons s1 (swapper s1 s2 (cdr slst))))
              (else (cons (car slst) (swapper s1 s2 (cdr slst))))))
      (else (cons (swapper s1 s2 (car slst))
                  (swapper s1 s2 (cdr slst)))))))
```

7. In this problem, `rotate` is the main function, and `last` and `rest` are helper functions.

```
(define last
        (lambda (los)
           (if (null? los) '()
               (list-ref los (- (length los) 1)))))
(define rest
        (lambda (los)
           (if (null? los) '()
               (if (null? (cdr los)) '()
                   (cons (car los) (rest (cdr los)))))))
(define rotate
        (lambda (los)
           (if (null? los) '()
               (cons (last los) (rest los)))))
```

8.

Write the append function in CPS.

9.

```
(define flatten
  (lambda (slst)
     (if (null? slst) '()
         (if (symbol? (car slst))
             (cons (car slst)(flatten (cdr slst)))
             (append (flatten (car slst)) (flatten (cdr slst)))))))
```