**;Currying and procedural abstraction**

(define **member?** ; original definition

(lambda (item ls)

(if (null? ls)

#f

(or (eq? (car ls) item)

(member? (cdr ls))))))

(define **member?**  ; curried version

(lambda (item ls)

((member?-c item) ls)))

(define **member?-c**

(lambda (item)

(letrec ([helper

(lambda (ls)

(if (null? ls)

#f

(or (eq? (car ls) item)

(helper (cdr ls)))))])

helper)))

(define **list-sum**

(letrec

([helper

(lambda (ls)

(if (null? ls)

0

(+ (car ls) (helper (cdr ls)))))])

helper))

(define **map** ; original definition

(lambda (proc ls)

(if (null? ls)

'()

(cons (proc (car ls) (map (cdr ls)))))))

(define **map** ;curried version

(lambda (proc ls)

((apply-to-all proc) ls)))

(define **apply-to-all**

(lambda (proc)

(letrec

([helper

(lambda (ls)

(if (null? ls)

'()

(cons (proc (car ls))

(helper (cdr ls)))))])

helper)))

(define **list-product**

(letrec

([helper

(lambda (ls)

(if (null? ls)

1

(\* (car ls) (helper (cdr ls)))))])

helper))

(define **list-recur**

(lambda (base-value list-proc)

(letrec

([helper

(lambda (ls)

(if (null? ls)

base-value

(list-proc (car ls) (helper (cdr ls)))))])

helper)))