```scheme
(define (null-ld? obj)
    (if (not (pair? obj)) #f (eq? (car obj) (cdr obj))))

(define (ld? obj)
    (cond
        ((null-ld? obj) #t)
        ((or (null? obj) (not (pair? obj)) (not (pair? (car obj)))) #f)
        (else (ld? (cons (cdr (car obj)) (cdr obj))))))

(define (cons-ld obj listdiff)
    (cons (cons obj (car listdiff)) (cdr listdiff)))

(define (car-ld listdiff)
    (car (car listdiff)))

(define (cdr-ld listdiff)
        (cons (cdr (car listdiff)) (cdr listdiff)))

(define (ld obj . more_listdiffs)
    (cons (cons obj more_listdiffs) null))

(define (length-ld listdiff)
    (let m_len ((m_ld listdiff) (len 0))
        (if (null-ld? m_ld) len (m_len (cdr-ld m_ld) (+ 1 len)))))

(define (append-ld listdiff . more_listdiffs)
    (if (null? more_listdiffs)
        listdiff
        (apply append-ld
            (cons (append
                (take (car listdiff) (length-ld listdiff))
                (car (car more_listdiffs)))
            (cdr (car more_listdiffs)))
            (cdr more_listdiffs))))

(define (ld-tail listdiff k)
    (if (equal? k 0)
        listdiff
        (ld-tail (cdr-ld listdiff) (- k 1))))

(define (list->ld list)
    (cons list '()))


(define (ld->list listdiff)
    (take (car listdiff) (length-ld listdiff)))

(define (map_list proc listdiff)
    (if (null-ld? listdiff)
        listdiff
        (cons-ld (proc (car-ld listdiff)) (map_list proc (cdr-ld listdiff)))))

(define (map-ld proc . more_listdiffs)
    (if (null? more_listdiffs)
        (cons '() '())
        (cons-ld (map_list proc (car more_listdiffs)) (apply map-ld proc (cdr more_listdiffs)))))

(define (expr2ld expr)
    (cond    [(not(pair? expr)) expr]
            [(null? (car expr)) (expr2ld (cdr expr))]
            [(equal? 'null? (car expr)) (cons 'null-ld? (expr2ld (cdr expr)))]
            [(equal? 'list? (car expr)) (cons 'ld? (expr2ld (cdr expr)))]
            [(equal? 'cons (car expr)) (cons 'cons-ld (expr2ld (cdr expr)))]
            [(equal? 'car (car expr)) (cons 'car-ld (expr2ld (cdr expr)))]
            [(equal? 'cdr (car expr)) (cons 'cdr-ld (expr2ld (cdr expr)))]
            [(equal? 'list (car expr)) (cons 'ld (expr2ld (cdr expr)))]
            [(equal? 'length (car expr)) (cons 'length-ld (expr2ld (cdr expr)))]
            [(equal? 'append (car expr)) (cons 'append-ld (expr2ld (cdr expr)))]
            [(equal? 'list-tail (car expr)) (cons 'ld-tail (expr2ld (cdr expr)))]
            [(equal? 'map (car expr)) (cons 'map-ld (expr2ld (cdr expr)))]
            [else (cons (expr2ld (car expr)) (expr2ld (cdr expr)))]
    )
)
```