```
(define (evalquote fn x)
  (apply fn x '()))

(define (apply fn x a)
  (cond ((atom fn) (cond ((eq? fn 'car) (caar x))
                         ((eq? fn 'cdr) (cdar x))
                         ((eq? fn 'cons) (cons (car x) (cadr x)))
                         ((eq? fn 'atom) (atom (car x)))
                         ((eq? fn 'eq) (eq? (car x) (cadr x)))
                         (else (apply (eval fn a) x a))))
        ((eq? (car fn) 'lambda) (eval (caddr fn) (pairlis (cadr fn) x a)))
        ((eq? (car fn) 'label) (apply (caddr fn) x
                                      (cons (cons (cadr fn) (caddr fn))
a)))))

(define (eval e a)
  (cond ((atom e) (cdr (assoc e a)))
        ((atom (car e)) (cond ((eq? (car e) 'quote) (cadr e))
                              ((eq? (car e) 'cond) (evcon (cdr e) a))
                              (else (apply (car e) (evlis (cdr e) a) a))))
        (else (apply (car e) (evlis (cdr e) a) a))))

; ------------------------------------------------------------------------

(define (evcon c a)
  (cond ((eval (caar c) a) (eval (cadar c) a))
        (else (evcon (cdr c) a))))

(define (evlis m a)
  (cond ((eq? m nil) nil)
        (else (cons (eval (car m) a) (evlis (cdr m) a)))))

; ------------------------------------------------------------------------

(define (pairlis keys vals alist)
  (if (eq? keys nil)
      alist
      (cons (cons (car keys) (car vals))
            (pairlis (cdr keys) (cdr vals) alist))))

(define (assoc x a)
  (cond ((eq? x (caar a)) (car a))
        (else (assoc x (cdr a)))))

; ------------------------------------------------------------------------

(evalquote '(lambda (x y) (cons (car x) y)) '((a b) (c d)))   ; = '(a c d)
```