Hand Evaluation  Using hand-evaluation, determine the meaning of these expressions. Show your steps.

For each step, copy and paste all of the definitions, changing any that are set!, to show how they change.

(begin (define f 1)
  (set! f (+ f 1))
  f)

Solution

(let ([f 1])
  (begin
    (define (inside)
      (begin
        (set! f (+ f 1))
        f)))
  (define f 2)
  2
  (begin
    (set! f (+ f 1))
    f)

Solution

(let ([f 1])
  (begin
    (set! f (+ f 1))
    f))
  (define f 3)
  2
  (begin
    f)

Solution

(let ([f 1])
  (begin
    (set! f (+ f 1))
    f))
  (inside)
  (inside)
Add to phone book  A phone-book is either:
  • '(), or
  • (cons (make-entry name phone-number) phone-book)

where name is a symbol and phone-number is a number.

(define-struct entry (name phone-number))

This is the template for phone-books:

(define (phone-book-template a-pb)
  (cond
Write this function:

;; adds name and number to a-pb unless the
;; name is already in the phone book
(define (add-to-phone-book a-pb name number)
  ...
)

Do not forget to design test cases before designing the function.

Solution

(define (add-to-phone-book a-pb name number)
  (cond
    [(null? a-pb) (cons (make-entry name number) '())]
    [else (if (eq? name (entry-name (car a-pb)))
               a-pb
               (cons (car a-pb)
                     (add-to-phone-book (cdr a-pb)
                                         name
                                         number)))]
  )

(add-to-phone-book '() 'me 5551212)
(cons (make-entry 'me 5551212) '())

(add-to-phone-book (cons (make-entry 'me 5551212) '()) 'me 5551212)
(cons (make-entry 'me 5551212) '())

(add-to-phone-book (cons (make-entry 'me 5551212) '()) 'you 5551234)
(cons (make-entry 'you 5551234)
       '()))

Lookup in phone book  Write this function:

;; lookup-in-phone-book : phone-book symbol → number or #f
;; looks up name in a-pb. Returns the name’s
;; number or #f if the name isn’t in the phone book.
(define (lookup-in-phone-book a-pb name)
  ...
)

Do not forget to design test cases before designing the function.

Solution

(define (lookup-in-phone-book a-pb name)
  (cond
    [(null? a-pb) #f]
    [else (if (eq? name (entry-name (car a-pb)))
               (entry-phone-number (car a-pb))
               (lookup-in-phone-book (cdr a-pb) name))]))
Adding State  Add a single definition for the current phone book, initially an empty phone book. Then, define the functions add and lookup:

```scheme
;; add : number symbol → void
;; adds name and number to the current phone book.
(define (add name number) . . .
)

;; lookup : symbol → number or #f
;; looks up name in the phone book
(define (lookup name) . . .
)
```

Solution

```scheme
(define current-phone-book '())
(define (add name number)

(define (lookup name)
```

Here are some tests for those functions:

```scheme
(lookup 'me) ;; should be
#f

(add 'me 5551212)
(lookup 'me)
;; should be
5551212
```