Report of Structure and Interpretation of Computer Programs

Chonbuk National University
Department of Computer Science
201514768
임유택

Problem 1

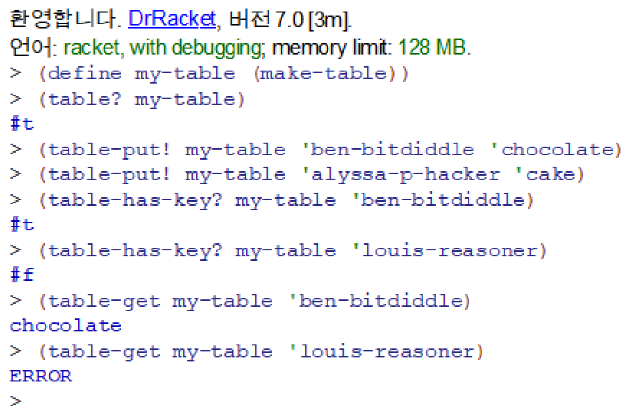
code;

```
#lang racket

;; By default, Racket doesn't have set-car! and set-cdr! functions.  The
;; following line allows us to use those:
(require r5rs)
(require rackunit)
;; Unfortunately, it does this by making cons return a "mutable pair"
;; instead of a "pair", which means that many built-ins may fail
;; mysteriously because they expect a pair, but get a mutable pair.
;; Re-define a few common functions in terms of car and friends, which
;; the above line make work with mutable pairs.
(define first car)
(define rest cdr)
(define second cadr)
(define third caddr)
(define fourth cadddr)
;; We also tell DrRacket to print mutable pairs using the compact syntax
;; for ordinary pairs.
(print-as-expression #f)
(print-mpair-curly-braces #f)
```

```
(define (find-assoc key table)
   (cond
      ((null? table) 'ERROR)
      ((equal? key (caar table)) (cadar table))
      (else (find-assoc key (rest table)))))


(define (add-assoc key val alist)
   (cons (list key val) alist))

(define table-tag 'table)

(define (make-table) (cons table-tag null))

(define (table? table1)
   (if (equal? (first table1) 'table)
         #t
         #f))

(define (table-get tbl key)
   (if (table? tbl)
         (find-assoc key (rest tbl))
         #f))

(define (table-put! tbl key val)
   (if (table? tbl)
         (set-cdr! tbl (add-assoc key val (rest tbl)))
         #f))
```

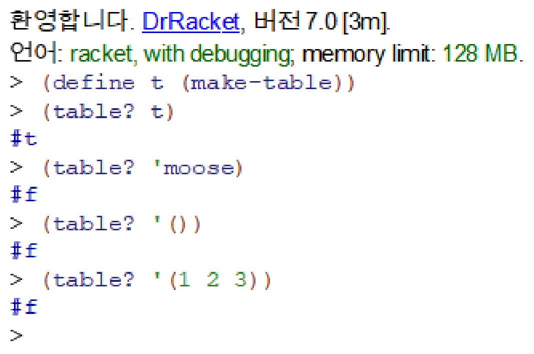
```
(define (table-has-key? tbl key)
   (if (table? tbl)
         (fortf-find-assoc key (rest tbl))
         #f))

(define (fortf-find-assoc key table)
         #f(cond
               [(null? table) #f]
               [(eq? key (caar table)) #t]
               [else (fortf-find-assoc key (rest table))]))

;; Allow this file to be included from elsewhere, and export all defined
;; functions from it.
(provide (all-defined-out))
```

Problem1



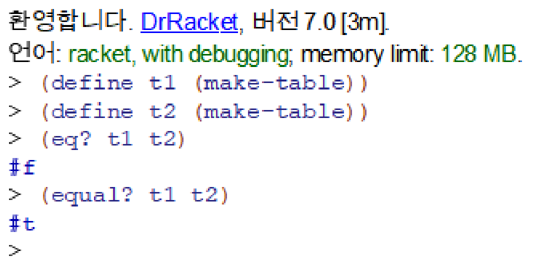
```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (define my-table (make-table))
> (table? my-table)
#t
> (table-put! my-table 'ben-bitdiddle 'chocolate)
> (table-put! my-table 'alyssa-p-hacker 'cake)
> (table-has-key? my-table 'ben-bitdiddle)
#t
> (table-has-key? my-table 'louis-reasoner)
#f
> (table-get my-table 'ben-bitdiddle)
chocolate
> (table-get my-table 'louis-reasoner)
ERROR
> 
```

test-case1



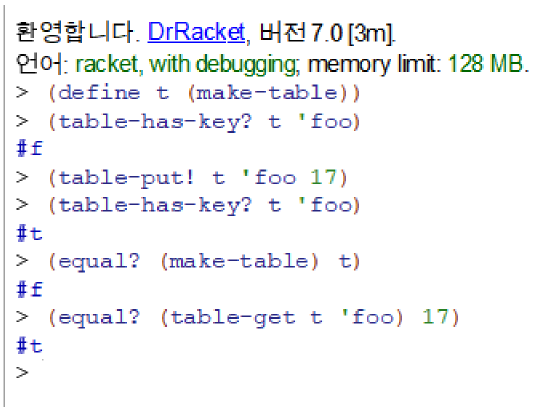
```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (define t (make-table))
> (table? t)
#t
> (table? 'moose)
#f
> (table? '())
#f
> (table? '(1 2 3))
#f
> 
```

test-case2



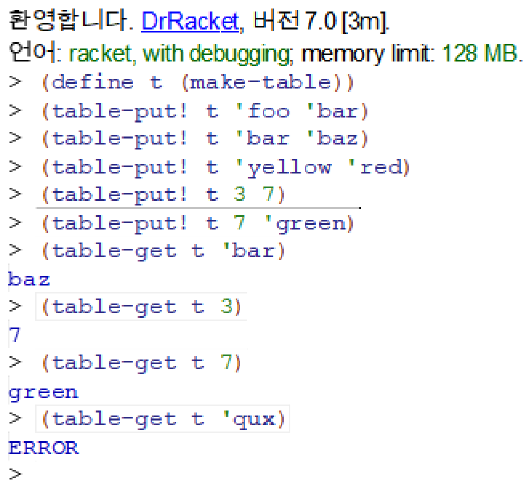
```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (define t1 (make-table))
> (define t2 (make-table))
> (eq? t1 t2)
#f
> (equal? t1 t2)
#t
> 
```

test-case3



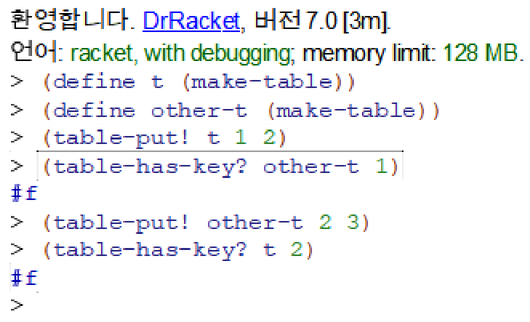
```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (define t (make-table))
> (table-has-key? t 'foo)
#f
> (table-put! t 'foo 17)
> (table-has-key? t 'foo)
#t
> (equal? (make-table) t)
#f
> (equal? (table-get t 'foo) 17)
#t
> 
```

test-case4



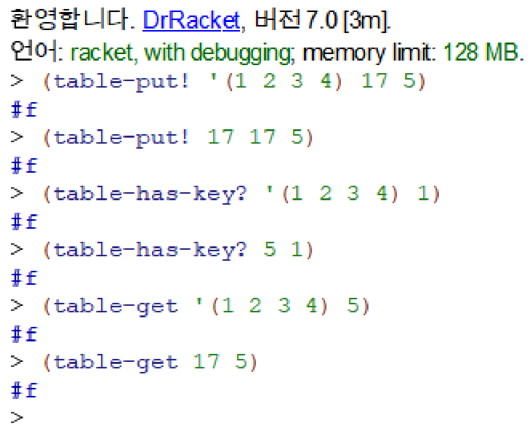
```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (define t (make-table))
> (table-put! t 'foo 'bar)
> (table-put! t 'bar 'baz)
> (table-put! t 'yellow 'red)
> (table-put! t 3 7)
> (table-put! t 7 'green)
> (table-get t 'bar)
baz
> (table-get t 3)
7
> (table-get t 7)
green
> (table-get t 'qux)
ERROR
>
```

test-case5



```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (define t (make-table))
> (define other-t (make-table))
> (table-put! t 1 2)
> (table-has-key? other-t 1)
#f
> (table-put! other-t 2 3)
> (table-has-key? t 2)
#f
>
```

test-case6



```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (table-put! '(1 2 3 4) 17 5)
#f
> (table-put! 17 17 5)
#f
> (table-has-key? '(1 2 3 4) 1)
#f
> (table-has-key? 5 1)
#f
> (table-get '(1 2 3 4) 5)
#f
> (table-get 17 5)
#f
>
```

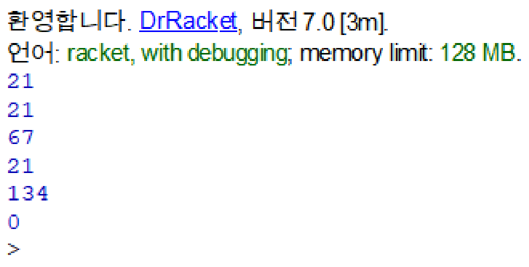
Problem 2

code;

```
#lang racket



(define (fib n)

   (if (< n 2)

         n

         (+ (fib (- n 1)) (fib (- n 2)))))



(define callnum 0)



(define (make-monitored fib)

   (lambda (n)

   (cond [(equal? n 'how-many-calls?) callnum]

            [(equal? n 'reset-call-count) (set! callnum 0)]

            [else

              (set! callnum (+ callnum 1))

              (if (< n 2)

                    n

                    (+ ((make-monitored fib) (- n 1)) ((make-monitored fib) (- n 2))))])))



(fib 8)

(set! fib (make-monitored fib))

(fib 8)

(fib 'how-many-calls?)
```

```
(fib 8)

(fib 'how-many-calls?)

(fib 'reset-call-count)

(fib 'how-many-calls?)
```

Problem2



```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
21
21
67
21
134
0
>
```

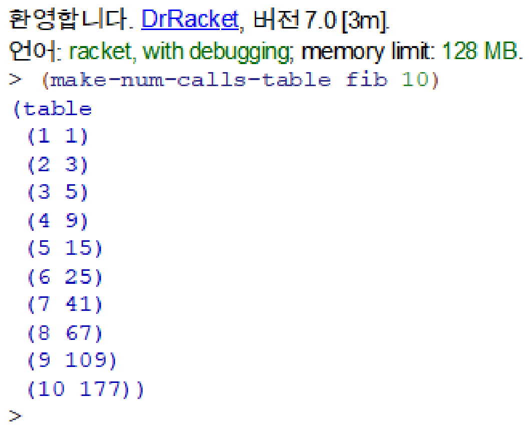
Problem 3

code;

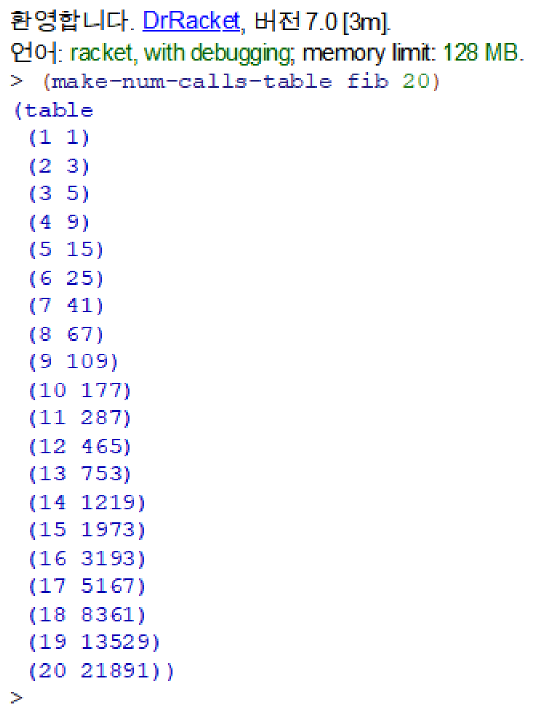
```
#lang racket


;; By default, Racket doesn't have set-car! and set-cdr! functions.  The

;; following line allows us to use those:

(require r5rs)

(require rackunit)

;; Unfortunately, it does this by making cons return a "mutable pair"

;; instead of a "pair", which means that many built-ins may fail

;; mysteriously because they expect a pair, but get a mutable pair.

;; Re-define a few common functions in terms of car and friends, which

;; the above line make work with mutable pairs.

(define first car)

(define rest cdr)

(define second cadr)

(define third caddr)
```

```
(define fourth cadddr)

;; We also tell DrRacket to print mutable pairs using the compact syntax

;; for ordinary pairs.

(print-as-expression #f)

(print-mpair-curly-braces #f)



(define table-tag 'table)


(define (make-table) (cons table-tag null))


(define table (make-table))


(define (add-assoc key val alist)

  (cons (list key val) alist))


(define (fib n)

  (if (< n 2)

      n

      (+ (fib (- n 1)) (fib (- n 2)))))


(define callnum 0)


(define (make-monitored fib)

  (lambda (n)

  (cond [(equal? n 'how-many-calls?) callnum]

        [(equal? n 'reset-call-count) (set! callnum 0)]

        [else

         (set! callnum (+ callnum 1))

         (if (< n 2)
              n
              (+ ((make-monitored fib) (- n 1)) ((make-monitored fib) (- n 2))))]])))


(define (make-num-calls-table fib n)
   ((make-monitored fib) n)
   (set-cdr! table (add-assoc n callnum (rest table)))
   (set! callnum 0)
   (if (< n 2)
         table
         (make-num-calls-table fib (- n 1))))



;; Allow this file to be included from elsewhere, and export all defined
;; functions from it.
(provide (all-defined-out))
```

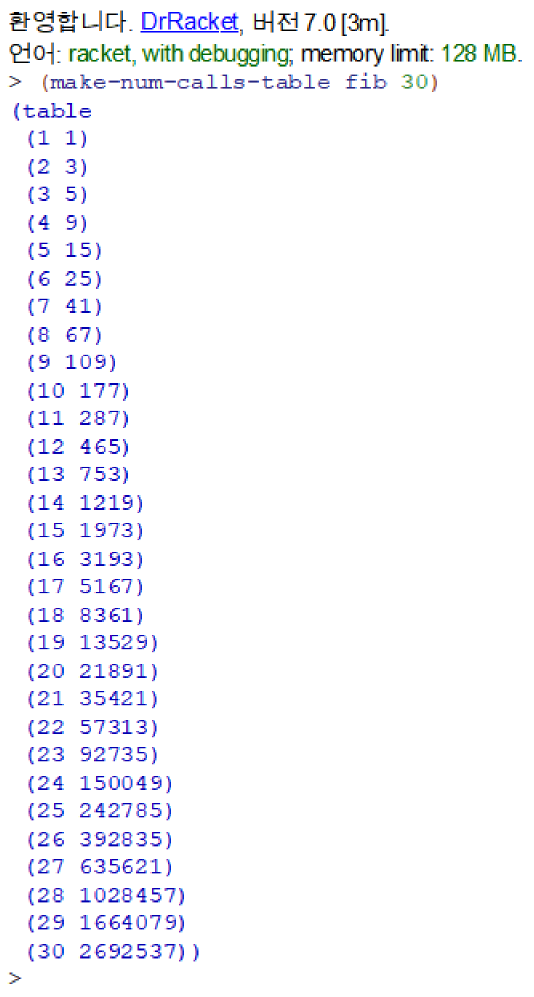
Problem3



```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (make-num-calls-table fib 10)
(table
 (1 1)
 (2 3)
 (3 5)
 (4 9)
 (5 15)
 (6 25)
 (7 41)
 (8 67)
 (9 109)
 (10 177))
>
```



```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (make-num-calls-table fib 20)
(table
 (1 1)
 (2 3)
 (3 5)
 (4 9)
 (5 15)
 (6 25)
 (7 41)
 (8 67)
 (9 109)
 (10 177)
 (11 287)
 (12 465)
 (13 753)
 (14 1219)
 (15 1973)
 (16 3193)
 (17 5167)
 (18 8361)
 (19 13529)
 (20 21891))
>
```



```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (make-num-calls-table fib 30)
(table
 (1 1)
 (2 3)
 (3 5)
 (4 9)
 (5 15)
 (6 25)
 (7 41)
 (8 67)
 (9 109)
 (10 177)
 (11 287)
 (12 465)
 (13 753)
 (14 1219)
 (15 1973)
 (16 3193)
 (17 5167)
 (18 8361)
 (19 13529)
 (20 21891)
 (21 35421)
 (22 57313)
 (23 92735)
 (24 150049)
 (25 242785)
 (26 392835)
 (27 635621)
 (28 1028457)
 (29 1664079)
 (30 2692537))
>
```

Problem 4

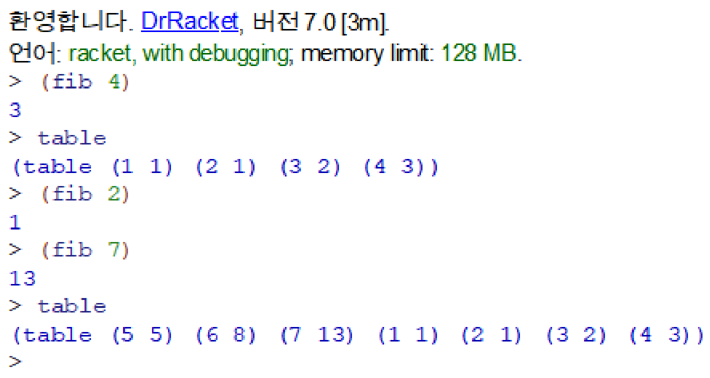
code;

```
#lang racket


;; By default, Racket doesn't have set-car! and set-cdr! functions.   The
;; following line allows us to use those:
(require r5rs)
(require rackunit)
;; Unfortunately, it does this by making cons return a "mutable pair"
;; instead of a "pair", which means that many built-ins may fail
;; mysteriously because they expect a pair, but get a mutable pair.
;; Re-define a few common functions in terms of car and friends, which
;; the above line make work with mutable pairs.
(define first car)
(define rest cdr)
(define second cadr)
(define third caddr)
(define fourth cadddr)
;; We also tell DrRacket to print mutable pairs using the compact syntax
;; for ordinary pairs.
(print-as-expression #f)
(print-mpair-curly-braces #f)



;;n을 입력하면 n까지의 rusursive를 이용한 fib값을 테이블에 put.
;;n보다 작은 수가 들어오면 recursive 이용이 아닌 table에서 get. (효율적)
;;n보다 큰 수 m이 들어오면 m부터 n까지 recursive를 이용해 fib값을 구하고 테이블에 put
;;지금까지 사용자가 입력한 수 보다 작은 값을 입력하면 table에서 가져와 비효율적인 recursive를 하지 않음.
```

```
(define table-tag 'table)

(define (make-table) (cons table-tag null))

(define table (make-table))

(define (add-assoc key val alist)
  (cons (list key val) alist))

(define (find-assoc key table)
  (cond
    ((null? table) 'ERROR)
    ((equal? key (caar table)) (cadar table))
    (else (find-assoc key (rest table)))))

(define (table-get tbl key)
  (if (table? tbl)
      (find-assoc key (rest tbl))
      #f))

(define (table? table1)
  (if (equal? (first table1) 'table)
      #t
      #f))

(define (table-has-key? tbl key)
  (if (table? tbl)
      (fortf-find-assoc key (rest tbl))
      #f))
```

```
(define (fortf-find-assoc key table)
        #f(cond
              [(null? table) #f]
              [(eq? key (caar table)) #t]
              [else (fortf-find-assoc key (rest table))]))

(define (fib n)
   (if (< n 2)
        n
        (+ (fib (- n 1)) (fib (- n 2)))))

(define (formemoizefib n)
   (if (< n 2)
        n
        (+ (formemoizefib (- n 1)) (formemoizefib (- n 2)))))

(define callnum 0)


(define (make-monitored fib)
   (lambda (n)
   (cond [(equal? n 'how-many-calls?) callnum]
            [(equal? n 'reset-call-count) (set! callnum 0)]
            [else
             (set! callnum (+ callnum 1))
             (if (< n 2)
                   n
                   (+ ((make-monitored fib) (- n 1)) ((make-monitored fib) (- n 2))))])))




(define (make-num-calls-table fib n)
   ((make-monitored fib) n)
   (set-cdr! table (add-assoc n callnum (rest table)))
   (set! callnum 0)
   (if (< n 2)
         table
         (make-num-calls-table fib (- n 1))))


(define room-num 0)

(define (search-and-add n table)
   (if (eq? (table-has-key? table n) #f)
         (set-cdr! table (add-assoc n (formemoizefib n) (rest table)))
         (table-get table n)))


(define (memoize fib)
   (lambda (n)
      (set! room-num 0)
      (set! room-num (+ room-num 1))
      (if (null? (rest table))
            (set-cdr! table (add-assoc n (formemoizefib n) (rest table)))
            (search-and-add n table))
      (if (< n 2)
            (table-get table room-num)
            ((running-memoize fib) (- n 1)))))


(define (running-memoize fib)
   (lambda (n)
      (set! room-num (+ room-num 1))

      (if (null? (rest table))

            (set-cdr! table (add-assoc n (formemoizefib n) (rest table)))

            (search-and-add n table))

      (if (< n 2)

            (table-get table room-num)

            ((running-memoize fib) (- n 1)))))




(set! fib (memoize fib))




;; Allow this file to be included from elsewhere, and export all defined

;; functions from it.

(provide (all-defined-out))
```

Problem4



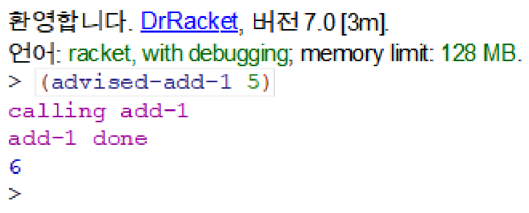
```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (fib 4)
3
> table
(table (1 1) (2 1) (3 2) (4 3))
> (fib 2)
1
> (fib 7)
13
> table
(table (5 5) (6 8) (7 13) (1 1) (2 1) (3 2) (4 3))
>
```

Problem 5

code;

```
#lang racket


(define (add-1 x) (+ x 1))


(define (advise func a b) (lambda(n) (a) (b) (func n)))


(define advised-add-1
   (advise add-1
             (lambda () (displayln "calling add-1"))
             (lambda () (displayln "add-1 done"))))
```

Problem5



```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (advised-add-1 5)
calling add-1
add-1 done
6
>
```

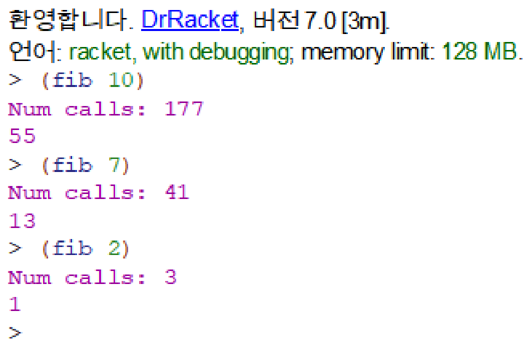
Problem 6

code;

```
#lang racket


;; By default, Racket doesn't have set-car! and set-cdr! functions.  The
;; following line allows us to use those:
(require r5rs)
(require rackunit)
;; Unfortunately, it does this by making cons return a "mutable pair"
;; instead of a "pair", which means that many built-ins may fail

;; mysteriously because they expect a pair, but get a mutable pair.

;; Re-define a few common functions in terms of car and friends, which

;; the above line make work with mutable pairs.

(define first car)

(define rest cdr)

(define second cadr)

(define third caddr)

(define fourth cadddr)

;; We also tell DrRacket to print mutable pairs using the compact syntax

;; for ordinary pairs.

(print-as-expression #f)

(print-mpair-curly-braces #f)



(define table-tag 'table)


(define (make-table) (cons table-tag null))


(define table (make-table))


(define (add-assoc key val alist)

   (cons (list key val) alist))


(define (find-assoc key table)

   (cond

      ((null? table) 'ERROR)

      ((equal? key (caar table)) (cadar table))

      (else (find-assoc key (cdr table)))))
```

```
(define (table-get tbl key)
  (if (table? tbl)
      (find-assoc key (cdr tbl))
      #f))

(define (table? table1)
  (if (equal? (car table1) 'table)
      #t
      #f))

(define (table-has-key? tbl key)
  (if (table? tbl)
      (fortf-find-assoc key (cdr tbl))
      #f))

(define (fortf-find-assoc key table)
      #f(cond
          [(null? table) #f]
          [(eq? key (caar table)) #t]
          [else (fortf-find-assoc key (cdr table))]))

(define (fib n)
  (if (< n 2)
      n
      (+ (fib (- n 1)) (fib (- n 2)))))

(define callnum 0)

(define (make-monitored fib)
  (lambda (n)
  (cond [(equal? n 'how-many-calls?) callnum]
        [(equal? n 'reset-call-count) (set! callnum 0)]
        [else
         (set! callnum (+ callnum 1))
         (if (< n 2)
             n
             (+ ((make-monitored fib) (- n 1)) ((make-monitored fib) (- n 2))))]))


(define (make-num-calls-table fib n)
  ((make-monitored fib) n)
  (set-cdr! table (add-assoc n callnum (cdr table)))
  (set! callnum 0)
  (if (< n 2)
      table
      (make-num-calls-table fib (- n 1))))



(define room-num 0)


(define (search-and-add n table)
  (if (eq? (table-has-key? table n) #f)
      (set-cdr! table (add-assoc n (fib n) (cdr table)))
      (table-get table n)))



(define (memoize fib)
  (set! room-num (+ room-num 1))
  (lambda (n)
    (if (null? (cdr table))
            (set-cdr! table (add-assoc n (fib n) (cdr table)))
            (search-and-add n table))
      (if (< n 2)
            (table-get table room-num)
      ((memoize fib) (- n 1)))))



(define (advise1 func fibnum a b c d) (a) (b) (c) (d))



(define (make-monitored-with-advice fib)
   (lambda (n)
      (set! callnum 0)
      (advise1 make-monitored
                  ((make-monitored fib) n)
                  (lambda () (display "Num calls: "))
                  (lambda () (display callnum))
                  (lambda () (display "₩n"))
                  (lambda () (display ((make-monitored fib) n)))
                  )))



 (set! fib (make-monitored-with-advice fib))



;; Allow this file to be included from elsewhere, and export all defined
;; functions from it.
(provide (all-defined-out))
```

Problem6


```
환영합니다. DrRacket, 버전 7.0 [3m].
언어: racket, with debugging; memory limit: 128 MB.
> (fib 10)
Num calls: 177
55
> (fib 7)
Num calls: 41
13
> (fib 2)
Num calls: 3
1
>
```