CS61A Fall 2003 Midterm 1, Clancy/Hilfinger

Problem 1 (6 points, 7 minutes)

Part a

Give a box-and-pointer diagram for ((A) B C).

Part b

Clearly fill in the parentheses and quotes so that evaluating the expression below produces the list ((A) B C) as a result.

```
  cons      A      B      C
```

Part c

Do the same for the following, in such a way that neither of append's arguments are empty.

```
  append      A      B      C
```

Problem 2 (12 points, 16 minutes)

Fill in the boxes in the following code so that it agrees with the comments. You may use the bigger procedure without defining it; don't use any other auxiliary procedures. You need no more than three cond clauses per box.

```
; L1 and L2 are lists of numbers.
; Return the list whose elements sum to the larger value.
(define (bigger L1 L2) ... )

;; L is a list of positive numbers; k is a number.
;; Return a subset of values in L that total most closely to k
;; without exceeding k, or #f if every subset of values in L
;; sums to a total greater than k.
;; Examples:
;; (best-subset '(6 4 5 3) -2) returns #f
;; (best-subset '(6 4 5 3) 2) returns ( )
;; (best-subset '(6 4 5 3) 8) returns (5 3)
;; (best-subset '(6 4 5 3) 9) returns either (6 3) or (4 5)
;; (best-subset '(6 4 5 3) 16) returns (6 4 5)
;; (best-subset '(6 4 5 3) 20) returns (6 4 5 3)

(define (best-subset L k)
  (cond
  ; base cases



    (else
      (let
        ((with-first (best-subset (cdr L) (- k (car L))))
         (without-first (best-subset (cdr L) k)))
        (cond



) ) ) ) )
```

Problem 3 (10 points, 20 minutes)

Here is a correctly working version of the 1extra? procedure you worked with in lab.

```
; Given lists big and small, return a true value when big is the result
; of inserting exactly one element into small, and return #f otherwise.
(define (1extra? big small)
  (helper '( ) big small) )
(define (helper part1 part2 L)
  (cond
    ((null? part2) #f)
    ((equal? (append part1 (cdr part2)) L) #t)
    (else (helper (append part1 (list (car part2))) (cdr part2) L)) ) )
```

Part a

List all the calls to helper, together with their arguments, that result from evaluating the expression

(1extra? '(D E F T U V) '(D E T U V))

Part b

Suppose B is the length of big and S is the length of small. Indicate *exactly* (i.e. *not* with a big-Theta estimate) how many calls to helper (including recursive calls) are produced in the worst case by a call to 1extra?, and briefly explain your answer.

For parts c and d, assume that

- append runs in time Theta(the length of its first arguments).
- car, cdr, and list run in time Theta(1).

Part c

Consider the condition

(equal? (append part1 (cdr part2)) L)

in helper. Suppose that, in the worst case, equal? runs in time Theta(the length of its shorter argument).

Let P1 be the length of part1, P2 the length of part2, and N the length of L, and assume that P1+P2=N+1 (that is, the arguments to equal? are the same length). Circle the expression below that most closely represents the worst-case running time to evaluate this condition once, and briefly explain your answer.

Theta(N^2)   Theta(N*P1)   Theta(P2)   Theta(P1)   Theta(N)

Brief explanation of your run-time estimate:

Part d

Now assume (as in part b) that B is the length of big and S is the length of small. Also assume that equal? runs in *constant* time. Circle the expression below that most closely represents the worst-case running time of 1extra? (with all calls to helper it produces), and briefly explain your answer.

Theta(S^3)   Theta(B^3)   Theta(S^2)   Theta(B^2)   Theta(B*S)   Theta(B+S)   Theta(S)   T

Brief explanation of your run-time estimate:

Problem 4 (10 points, 16 minutes)

Write a procedure named updated that is called as follows:

(updated table name new-value)

Arguments to updated are the following.

- Table is a list of two-element lists--table *entries*--each of which has a symbol as its first element and the associated value for that symbol as its second element. At most one entry in the table has a particular symbol as its first symbol so
((mike (senior lecturer)) (paul professor))

is a legal argument but

((mike cs61a) (paul cs61a) (mike (cs9 cs3s)))

is not (since there are two entries for mike).

- Name is a symbol.

  Updated should return a new table that's the same as table, except that the value associated with name should be new-value. If there is no entry in table for name, an entry should be added; if there is already an entry in table for name, its associated value should be *replaced*. The table returned by updated should have at most one entry for each name. The sequence of entries in the table is not important.

  Examples:

  (updated '((john 53) (jane 52)) 'maria 35)

  should return a three-element table containing (john 53), (jane 52), and (maria 35).

  (updated '((john 53) (jane 52)) 'jane 74)

  should return a two-element table containing (john 53) and (jane 74).