CS314 Spring 2023

Homework 5

Due Tuesday, April 4, 11:59pm

SAMPLE SOLUTION

submission: text file "hw5.ss" through canvas

Problem 1

Write Scheme programs that generate the following lists as output using only `cons` as the list building operator:

1.
```
;;  '(a (b c) d ((e f) (g)))
(cons 'a
   (cons (cons 'b (cons 'c '()))
       (cons 'd
           (cons
               (cons (cons 'e (cons 'f '()))
               (cons (cons 'g '()) '())) '()))))
```

2.
```
;;  '(* a 4) such that ((car '(* a 4)) 5 3) evalutes to 15
(cons * (cons 'a (cons 4 '())))
```

Problem 2 – Scheme

Write the following functions on lists in Scheme. The semantics of the functions is decribed through examples.

1.
```
(define flatten
  (lambda (l)
     (cond
        ((null? l) '())
        ((list? (car l)) (append (flatten (car l)) (flatten (cdr l))))
        (else (cons (car l) (flatten (cdr l)))))))

   ;; (flatten '(a ((b) (c d) (((e))))) --> '(a b c d e)
```

2.
```
(define rev
   (lambda (l)
      (cond
        ((null? l) '())
        ((list? (car l)) (append (rev (cdr l)) (cons (rev (car l)) '())))
```

```
          (else (append (rev (cdr l)) (cons (car l) '())))))))

      ;; (rev '(a((b)(c d)(((e))))))) --> '(((((e)))(d c)(b))a)
      ;; Note: Do not use the Scheme build-in function "reverse".
```

3.
```
(define delete
    (lambda (a l)
       (cond
         ((null? l) '())
         ((list? (car l)) (cons (delete a (car l))  (delete a (cdr l))))
         (else (if (eq? a (car l))
            (delete a (cdr l))
            (cons (car l) (delete a (cdr l)))))))))

      ;; (delete 'c '(a((b)(c d)(((e)))))) --> (a((b)(d)(((e)))))
      ;; (delete 'f '(a((b)(c d)(((e)))))) --> (a((b)(c d)(((e)))))
```

4.
```
(define merge-sorted
    (lambda (x y)
      (cond
        ((null? x) y)
        ((null? y) x)
        ((< (car x) (car y)) (cons (car x) (merge-sorted (cdr x) y)))
        ((eq? (car x) (car y)) (merge-sorted (cdr x) y))
        (else
            (cons (car y) (merge-sorted x (cdr y)))))))

      ;; lists x and y are sorted; no duplications in result list
      ;;(merge-sorted '(4 8 12 17 45) '(2 4 9 24)) --> '(2 4 8 9 12 17 24 45)
```

Problem 3

Implement a symbol table data type that supports the following operations:

1. `NewTable()` : returns an empty table value;
2. `InsertIntoTable((variable value), table)` : inserts a variable/value pair into the table;
3. `LookupTable(variable, table)` : finds entry for variable and returns its value. If no variable is found, the empty list is returned. If more than one entry for a variable, the most recently entered value for that variable will be returned.

```
(define NewTable
  (lambda () '()))

(define InsertIntoTable
  (lambda (entry table) ;; entry is a list of a variable and a value
     (cons entry table)))

(define LookupTable
  (lambda (variable table)
     (cond
       ((null? table) '())
       ((eq? variable (caar table)) (cadar table))
       (else
         (LookupTable variable (cdr table))))))

(define table
    (InsertIntoTable '(b (2 4 5)) (InsertIntoTable '(a 7) (NewTable))))

;;(LookupTable 'a table) --> 7
;;(LookupTable 'b table) --> '(2 4 5)
;;(LookupTable 'c table) --> '()
```

Problem 4

Use the `map` and `reduce` functions defined as

```
(define map
  (lambda (f l)
    (if (null? l)
        '()
```

```
        (cons (f (car l)) (map f (cdr l))) )))

(define reduce
  (lambda (op l id)
     (if (null? l)
        id
        (op (car l) (reduce op (cdr l) id)) )))
```

to implement functions `minSquareVal` and `maxSquareVal` that determine the minimal square value and maximal square value, respectively, of a list of integer numbers. Note: **Solution cannot accept an empty list as an actual argument**.

```
(define minSquareVal
  (lambda (l)
    (let ((sqrl (map (lambda (x) (* x x)) l))
          (initval (* (car l) (car l))))
      (reduce
       (lambda (x y) (if (< x y) x y))
       sqrl
       initval))))

(define maxSquareVal
  (lambda (l)
    (let ((sqrl (map (lambda (x) (* x x)) l))
          (initval (* (car l) (car l))))
      (reduce
       (lambda (x y) (if (> x y) x y))
       sqrl
       initval))))
```