Thomas Nguyen
COEN 171
Professor Vu

Assignment #9

Problem Sets

15.2 Find the definitions of Scheme functions *EVAL* and *APPLY*, and explain their actions.

EVAL

- Parameters:
    - an expression
    - an environment
- Detects the type of the expression and then evaluates it.

APPLY

- Parameters:
    - a function
    - an expression or list of arguments
- The function is applied to the expression, and is returned

15.8 If Scheme were a pure functional language, could it include *DISPLAY*? Why or why not?

If Scheme includes DISPLAY, it would no longer be a pure functional language because DISPLAY has the side effect of producing an output.

15.9 What does the following Scheme function do?

```
(define (y s lis)
   (cond
        ((null? lis) '() )
        ((equal? s (car lis)) lis)
        (else (y s (cdr lis)))
))
```

Parameters:

- *s*, an element
- *lis*, a list

Functionality:

- The function *y* returns a list of all elements up to the first occurrence of *s*

Programming Exercises

15.1 Write a Scheme function that computes the area of a circle, given its radius

```
define(pi 3.14159)
(define (sphere_volume radius)
    (* (/ 4 3) pi
        expt radius 3)
)
```

15.2 Write a Scheme function that computes the real roots of a given quadratic equation. If the roots are complex, the function must display a message indicating that. This function must use an IF function. The three parameters to the function are the three coefficients of the quadratic equation.

```
(DEFINE (quadratic_roots a b c)
    (LET (
        (term1 (/ (- 0 b)(* 2 a)))
        (term2 (/ (SQRT (- (* b b)(* 4 a c)))(* 2 a)))
        (LIST (+ term1 term2)) (- term1 term2))
    )
)
```

15.5 Write a Scheme function that returns the number of zeros in a given simple list of numbers.

```
(DEFINE (countzeros list)
    (COND
        ((null? lis) 0)
        ((= (car list) 0) (+ 1 countzeros (cdr list)))
        (ELSE (countzeros (cdr list)))
    )
)
```

15.6 Write a Scheme function that takes a simple list of numbers as a parameter and returns a list with the largest and smallest numbers in the input list.

```
(DEFINE (max min lis)
    (list (apply max lis)
    (apply min lis))
)
```

15.11 Write a Scheme function that returns the reverse of its simple list parameter.

```
(DEFINE (reverse lis)
    (COND
        ((NULL? lis) '())
        (ELSE (APPEND (reverse (CDR lis) () )))
    )
)
```

15.19 Write the quicksort algorithm in Scheme.

```
(DEFINE (partition compare li)
    (COND
        ((null? l1) '())
        (else (let ((pivot (car l1)))
            (append (append (quicksort (partition (lambda (x) (< x pivot)) l1)))
                (partition (lambda (x) (= x pivot)) l1)
            )
            (quicksort (partition (lambda (x) (> x pivot)) l1))
        ))
    )
)
```

15.20 Rewrite the following Scheme function as a tail-recursive function

```
(DEFINE (doitHelper n doit_)
    (IF (= n 0)
        doit
        (doitHelper (- n 1) (+ n doit))
    )
)

(DEFINE (doit n)
    (doitHelper n 0)
)
```