Problem Set

Question#9

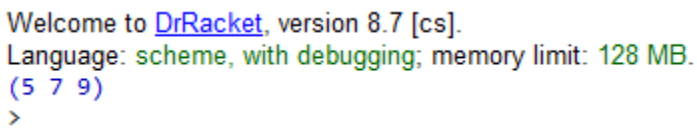
Theory

Function y has two parameters. First parameter is a variable and other is a list. This function runs till it matches the variable with the top element of list or list is empty. Then it will return a new list with all element removed till it matches the variable.

Code

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

(y 5 '(1 3 5 7 9))
```

Screenshot



Question#10

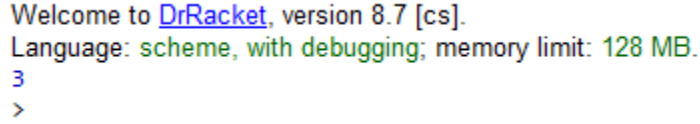
Theory

Function x has one parameter which is list. This function count all the elements of list. If the element is not #f or '() it will add 1. Then it returns the count excluding #f and '().

Code

```
(define (x lis)
  (cond
    ((null? lis) 0)
    ((not (list? (car lis)))
     (cond
       ((eq? (car lis) #f) (x (cdr lis)))
       (else (+ 1 (x (cdr lis))))))
    (else (+ (x (car lis)) (x (cdr lis)))))
  )

(x '(() 1 () 2 3 ()))
```

Screenshot



Programming Assignment

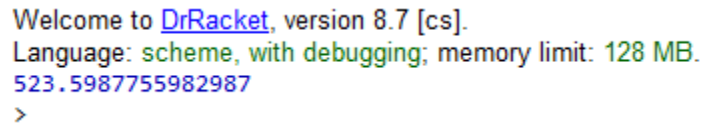
Question#1

Write a Scheme function that computes the volume of a sphere, given its radius.

Code

```
(define (volume_of_a_sphere r)
  (* (/ 4 3) (* pi (* r r r)))
  )

(volume_of_a_sphere 5)
```

Screenshot



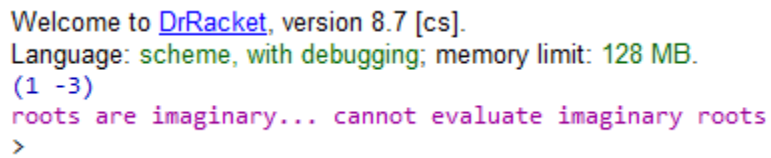
Question#2

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.

Code

```
(define (root_part a b c)
  (- (* b b) (* 4 a c)))

(define (root_part_underscore_2a a b c)
  (/ (sqrt (root_part a b c)) (* 2 a)))

(define (minus_b_underscore_2a a b c)
  (/ (- 0 b) (* 2 a)))

(define (quadratic_roots a b c)
  (if (< (root_part a b c) 0) (printf "roots are imaginary... cannot evaluate imaginary roots\n")
      (cons (+ (minus_b_underscore_2a a b c) (root_part_underscore_2a a b c))
            (cons (- (minus_b_underscore_2a a b c) (root_part_underscore_2a a b c)) '())))
  )

(quadratic_roots 2 4 -6)
(quadratic_roots 4 4 6)
```

Screenshot



Question#4

Write a Scheme function that takes two numeric parameters, A and B, and returns A raised to the B power.

Code

```
(define (^ A B)
  (if (= B 0) 1
      (if (< B 0)
          (/ (^ A (+ B 1)) A)
          (* A (^ A (- B 1))))
      )
  )

(^ 2 3)
(^ 2 -4)
(^ -2 3)
```

Screenshot

```
Welcome to DrRacket, version 8.7 [cs].
Language: scheme, with debugging; memory limit: 128 MB.
8
 1
16
-8
>
```

Question#5

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

Code

```
(define (num_of_zeros lis)
  (if (null? lis) 0
      (if (= 0 (car lis)) (+ 1 (num_of_zeros (cdr lis)))
          (num_of_zeros (cdr lis)))
      )
  )

(num_of_zeros '(1 0 1 0 1 1 0 1 0 0 1))
```

Screenshot

```
Welcome to DrRacket, version 8.7 [cs].
Language: scheme, with debugging; memory limit: 128 MB.
5
>
```

Question#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.

Code

```
(define (find_min lis)
  (cond
    ((null? lis) '())
    ((null? (cdr lis)) (car lis))
    ((< (car lis) (find_min (cdr lis))) (car lis))
    (else (find_min (cdr lis))))
```

```
  )
(define (find_max lis)
  (cond
    ((null? lis) '())
    ((null? (cdr lis)) (car lis))
    ((> (car lis) (find_max (cdr lis))) (car lis))
    (else (find_max (cdr lis))))
  )

(define (minmax lis)
   (cons (find_min lis) (cons (find_max lis) '()))
  )

(minmax '(1 3 5 7 9 2 4 6 8 10))
```

Screenshot

```
Welcome to DrRacket, version 8.7 [cs].
Language: scheme, with debugging; memory limit: 128 MB.
(1 10)
>
```

Question#7

Write a Scheme function that takes a list and an atom as parameters and returns a list identical to its parameter list except with all top-level instances of the given atom deleted

Code

```
(define (remove_identical lis s)
  (cond
    ((null? lis) '())
    ((eq? (car lis) s) (remove_identical (cdr lis) s))
    (else (cons (car lis) (remove_identical (cdr lis) s))))
  )

(remove_identical '(T A L H A) 'A)
```

Screenshot

```
Welcome to DrRacket, version 8.7 [cs].
Language: scheme, with debugging; memory limit: 128 MB.
(T L H)
>
```

TPL Assignment

Problem#1

Define a Scheme function flatten that flattens a given list. The function should work with general lists potentially having sublists as elements. For example, (flatten '(A B (C (D D) C) B A)) evaluates to (A B C D D C B A).

Code

```
(define (flatten lis)
  (cond
```

```
    ((null? lis) '())
    ((list? (car lis))
     (append (flatten (car lis))
             (flatten (cdr lis))))
    (else (cons (car lis) (flatten (cdr lis))))
    )
  )

(flatten '((A B (C (D D) C) B A)))
```

Screenshot

```
Welcome to DrRacket, version 8.7 [cs].
Language: scheme, with debugging; memory limit: 128 MB.
(A B C D D C B A)
>
```

Problem#2

Define a Scheme function slice to extract an (i; j)-slice from a given list, where i and j are two indices in the list. An (i; j)-slice of a list l is the list containing all the elements starting from the i'th element up to but not including the k'th element of l. Note that indexing should start from 0. The function slice should behave appropriately on unexpected values for its arguments. For example, (slice 2 4 '(A B C D E)) evaluates to (C D).

Code

```
(define (slice start end lis)
  (if (> start end) (printf "fatal error: start_offset > end_offset\n")

      (cond
        ((> start 0) (slice (- start 1) (- end 1) (cdr lis)))
        ((> end 0) (cons (car lis) (slice start (- end 1) (cdr lis))))
        (else '()))
      )
  )

(slice 2 4 '(A B C D E))
(slice 4 2 '(A B C D E))
```

Screenshot

```
Welcome to DrRacket, version 8.7 [cs].
Language: scheme, with debugging; memory limit: 128 MB.
(C D)
fatal error: start_offset > end_offset
>
```

Problem#3

Define a Scheme function lsort that, given a list l of lists, sorts the elements of l according to their length in ascending order; i.e. it produces a list in which shorter lists appear before longer lists in the result. Note that the order in which lists of the same length appear is not speci_ed. For example, (lsort '((A B C) (D E) (F G H) (D E) (I J K L) (M N) (O))) evaluates to ((O) (D E) (D E) (M N) (A B C) (F G H) (I J K L)).

Code

```
;https://stackoverflow.com/questions/72072765/sort-list-of-lists-by-length
(define (lsort ls)
```

```
  (sort ls (lambda (x y) (< (length x) (length y))))
  )

(lsort '((A B C) (D E) (F G H) (D E) (I J K L) (M N) (O)))
```

Screenshot

```
Welcome to DrRacket, version 8.7 [cs].
Language: scheme, with debugging; memory limit: 128 MB.
((O) (D E) (D E) (M N) (A B C) (F G H) (I J K L))
>
```

Problem#4

Define a Scheme function gcd that computes the greatest common divisor for two positive integers given as arguments. For example, (gcd 52 108) evaluates to 2.

Code

```
(define (gcd lhs rhs)
  (cond
    ((> lhs rhs) (gcd (- lhs rhs) rhs))
    ((< lhs rhs) (gcd lhs (- rhs lhs)))
    (else lhs))
  )

(gcd 52 108)
```

Screenshot

```
Welcome to DrRacket, version 8.7 [cs].
Language: scheme, with debugging; memory limit: 128 MB.
4
>
```

Problem#5

Define a Scheme function prime-factors that constructs a list containing the prime factors in ascending order of a given positive integer. For example, (prime-factors 315) evaluates to (3 3 5 7).

Code

```
(define (loop n i)
  (if (<= n 1) '()
      (if (zero? (remainder n i)) (cons i (loop (/ n i) i))
          (loop n (+ i 1))))
  )

(define (prime-factors n) (loop n 2))

(prime-factors 315)
```

Screenshot

```
Welcome to DrRacket, version 8.7 [cs].
Language: scheme, with debugging; memory limit: 128 MB.
(3 3 5 7)
>
```

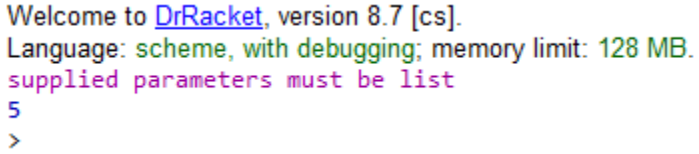
Problem#6

Define a function to compute the length of a list.

Code

```
(define (compute_len lis)
  (if (list? lis)
      (if (null? lis) 0
          (+ 1 (compute_len (cdr lis))))
      (printf "supplied parameters must be list\n"))
  )

(compute_len 1)
(compute_len '(1 2 3 4 5))
```

Screenshot



Problem#7

Define a function to compute sum of squares of number of the list

Code

```
(define (sum_of_square lis)
  (if (list? lis)
      (if (null? lis) 0
          (+ (* (car lis) (car lis)) (sum_of_square (cdr lis))))
      (printf "supplied parameters must be list\n"))
  )

(sum_of_square 1)
(sum_of_square '(1 3 5 7 9))
```

Screenshot