#1

```
1  ; Taiki Tsukahara
2  ; Team Members - Aaron Garcia, Taiki Tsukahara, Nhat Minh Dinh
3  ; Lab 4 Due 11/24/2023
4  ; Problem 1
5  #lang scheme
6  (define log2
7    (lambda (n) ;input n
8      (letrec
9          ((helper
10           (lambda (n acc)
11             (if (= n 1) ; if input is 1
12                 acc ; return the result
13                 (helper (quotient (+ n 1) 2) (+ acc 1)))))) ; recursive call
14
15       (helper n 0)))) ; initial values
16
17 (display "Log base 2 of 4 is ")
18 (display (log2 4)) ; what is the log base 2 of 4 = 2
19
```

```
Welcome to DrRacket, version 8.11 [cs].
Language: scheme, with debugging; memory limit: 512 MB.
Log base 2 of 4 is 2
>
```

#2

```
1  ; Taiki Tsukahara
2  ; Team Members - Aaron Garcia, Taiki Tsukahara, Nhat Minh Dinh
3  ; Lab 4 Due 11/24/2023
4  ; Problem 2
5  #lang scheme
6  (define filter
7    (lambda (f L)
8      ; return list of those elements in L which pass through filter f
9      (if (null? L) L ; if list L is empty?
10         (if (f (car L)) ; checks first element of L
11             (cons (car L) (filter f (cdr L))) ; makes new list by combining 2 parts.
12             (filter f (cdr L)))))) ; recursive call
13
14 (define input '(3 9 5 8 2 4 7)) ; input value here
15 (display "From the list:(3 9 5 8 2 4 7)")
16 (newline)
17 (display "Number less than 5 are ")
18 (display (filter (lambda (num) (< num 5)) input)) ; Output: (3 2 4)
19
```

```
Welcome to DrRacket, version 8.11 [cs].
Language: scheme, with debugging; memory limit: 512 MB.
From the list:(3 9 5 8 2 4 7)
Number less than 5 are (3 2 4)
>
```

#3

```
1  ; Taiki Tsukahara
2  ; Team Members - Aaron Garcia, Taiki Tsukahara, Nhat Minh Dinh
3  ; Lab 4 Due 11/24/2023
4  ; Problem 3
5  #lang scheme
6  (define rotations
7    (lambda (L)
8      (letrec
9          ((helper
10           (lambda (Ls A B)
11             (if (null? B)
12                 Ls
13                 (helper (append Ls (list (append B A)))
14                         (append A (list (car B)))
15                         (cdr B))))))
16       (helper '() '() L))))
17 (display "Rotations of (a b c d e):")
18 (newline)
19 (display (rotations '(a b c d e)))
20 ; Output: ((a b c d e) (b c d e a) (c d e a b) (d e a b c) (e a b c d))
```

```
Welcome to DrRacket, version 8.11 [cs].
Language: scheme, with debugging; memory limit: 512 MB.
Rotations of (a b c d e):
((a b c d e) (b c d e a) (c d e a b) (d e a b c) (e a b c d))
>
```

#4

```
1  ; Taiki Tsukahara
2  ; Team Members - Aaron Garcia, Taiki Tsukahara, Nhat Minh Dinh
3  ; Lab 4 Due 11/24/2023
4  ; Problem 4
5  #lang scheme
6
7  (define reverse
8    (lambda (lst)
9      (cond
10       ((null? lst) '()) ; If list is empty = return an empty list
11       (else (append (reverse (cdr lst)) (list (car lst)))))))
12 ; Reverse the rest and append the first element at the end
13
14 (define display-reverse
15   (lambda (lst)
16     (display "(")
17     (let loop ((lst lst))
18       (cond
19         ((null? lst) (display ")"))
20         ((null? (cdr lst)) (display (car lst)) (display ")"))
21         (else (display (car lst)) (display " ") (loop (cdr lst)))))))
22
23 (define myinput '(1 1 2 A 2 B 3 C)) ; input value here
24 (display "Reverse order of ")
25 (display myinput)
26 (display " is ")
27 (display-reverse (reverse myinput)) ; (1 1 2 A 2 B 3 C) = (C 3 B 2 A 2 1 1)
28
```

```
Welcome to DrRacket, version 8.11 [cs].
Language: scheme, with debugging; memory limit: 512 MB.
Reverse order of (1 1 2 A 2 B 3 C) is (C 3 B 2 A 2 1 1)
>
```