```
#lang racket
 1
 2
 3
    (struct empty-list () #:transparent)
 4
 5
    (define empty (empty-list))
    empty
 6
 7
 8
    (struct cons-list (v prev) #:transparent)
 9
10
    (cons-list 5 empty)
11
12
    :(cons-list 5 5)
13
14
    (define (make-list args)
      (match args
15
        ((list)(empty-list))
16
        ((list i (? empty-list? x))(cons-list i x))
17
18
        ((list i (? cons-list? x))(cons-list i x))
19
        ((list i)(cons-list i (empty-list)))
        (_ (printf "Error: not a valid list!\n")))
20
21
22
    (make-list (list ))
23
    (make-list (list 1))
    (make-list (list 2 empty))
24
    (make-list (list 1 2 3 4))
25
26
27
    (define (get-first-list lst)
28
      (match lst
        ((? empty-list? lst) (printf "Error: not
29
    long enough!\n"))
29
        ((cons-list v rest) v)))
30
31
32
    (define alpha (make-list (list "c" (make-list
    (list "b" (make-list (list "a")))))))
32
33
    alpha
```

```
34
    (get-first-list alpha)
35
36
    (define (get-rest-list lst)
37
      (match lst
        ((? empty-list? lst) (printf "Error: not
38
38
    long enough!\n"))
        ((cons-list v rest) rest)))
39
40
41
    (get-rest-list alpha)
42
43
    (define (get-item-index lst i)
44
      (letrec ((helper (lambda (lst i j)
                          (if (= i j)
45
46
                               (get-first-list lst)
47
                               (helper (get-rest-list
47
    lst) i (+ j 1))))))
48
        (helper lst i 0)))
49
    (get-item-index alpha 1)
50
51
    (define (length-list lst)
52
      (cond ((empty-list? lst) 0)
53
             ((cons-list? lst) (+ 1)
54
                                   (length-list
54
    (get-rest-list lst))))
             (else (printf "Error: not a list!"))))
55
56
57
    (length-list alpha)
58
59
    (define (length-list-1 lst)
      (letrec ((helper (lambda (l len)
60
61
                          (if (empty-list? l)
62
                               len
                               (helper (get-rest-list
63
63
    1)
64
                                       (+ 1 len)))))
```

```
(helper lst 0)))
65
66
67
    (length-list-1 alpha)
68
69
    (define (length-list-2 lst)
      (match lst
70
71
        ((empty-list) 0)
72
        ((cons-list _ rest) (+ 1 (length-list-2)
72
    rest)))
73
        (_ "Error: not a list!")))
74
    (length-list-2 alpha)
75
```