

```
1 #lang racket
2
3 (define (sum-numbers lst)
4   (if (= (length lst) 1)
5       (first lst)
6       (+ (first lst)
7          (sum-numbers (rest lst)))))
8
9 (define numbers (list 1 2 3 4 5))
10
11 (sum-numbers numbers)
12
13 (define (append-strings lst)
14   (if (= (length lst) 1)
15       (first lst)
16       (string-append (first lst)
17                       (append-strings (rest lst)))))
18
19 (define animals (list "cat" "penguin" "hedgehog"))
20
21 (append-strings animals)
22
23 (define (first-fold f lst)
24   (if (= (length lst) 1)
25       (first lst)
26       (f (first lst)
27          (first-fold f (rest lst)))))
28
29 (first-fold (lambda (x y)(+ x y)) numbers)
30
31 (first-fold (lambda (x y)(string-append x y)) animals)
32
33 (define (is-all-numbers? lst)
34   (if (= (length lst) 1)
35       (number? (first lst))
36       (and (number? (first lst))
37            (is-all-numbers? (rest lst)))))
38
39 (is-all-numbers? (list #t 5 "cat"))
40 (is-all-numbers? numbers)
41
42 (define (sum-numbers-2 lst)
43   (letrec ((helper (lambda (l acc)
44                     (if (empty? l)
45                         acc
46                         (helper (rest l) (+ (first l) acc)))))))
```

```

47     (helper lst 0)))
48
49 (sum-numbers-2 numbers)
50
51 (define (is-all-numbers?-2 lst)
52   (letrec ((helper (lambda (l acc)
53                     (if (empty? l)
54                         acc
55                         (helper (rest l) (and (number? (first
56 l))
57                                         acc))))))
58     (helper lst #t)))
59 (is-all-numbers?-2 numbers)
60
61 (define (my-fold f acc lst)
62   (if (empty? lst)
63       acc
64       (my-fold f
65               (f (first lst)
66                 acc)
67               (rest lst))))
68
69 (my-fold (lambda (x acc) (string-append x acc)) "" animals)
70
71 ;; built-in fold
72
73 (define (sum-numbers-foldl lst)
74   (foldl (lambda (x y)(+ x y)) 0 lst))
75
76 (sum-numbers-foldl numbers)
77
78 (define (sum-numbers-foldr lst)
79   (foldr (lambda (x y)(+ x y)) 0 lst))
80
81 (sum-numbers-foldr numbers)
82
83 (define (append-strings-l lst)
84   (foldl (lambda (x y)(string-append x y)) "" lst))
85
86 (append-strings-l animals)
87
88 (define (append-strings-r lst)
89   (foldr (lambda (x y)(string-append x y)) "" lst))
90
91 (append-strings-r animals)

```

```
92
93 ;;
94
95 (foldl (lambda (x y z) (string-append x y z)) "" animals animals)
96
97 (foldl (lambda (x y)(string-append x y)) "" (map (lambda (x)
97 (string x))
98                                     (string->list
98 "animals")))
99
100
101 (foldl (lambda (x y)(string-append (string x) y)) ""
101 (string->list "animals"))
102
103
104
105
```