

```
1 #lang racket
2
3 (define (add-5 lst)
4   (if (equal? lst null)
5       null
6       (cons (+ (first lst) 5)
7             (add-5 (rest lst)))))
8
9 (add-5 (list 1 2 4))
10
11 (define (countdown n)
12   (println n)
13   (if (= n 0)
14       (void)
15       (countdown (- n 1))))
16
17 (countdown 10)
18
19 (define (count-help curr end)
20   (println curr)
21   (if (= curr end)
22       (void)
23       (count-help (+ curr 1) end)))
24
25 (count-help 0 4)
26
27 (define (countup n)
28   (count-help 0 n))
29
30 (countup 4)
31
32 (define (countup-2 n)
33   (letrec ((count-help-2 (lambda (curr end)
34                             (println curr)
35                             (if (= curr end)
36                                 (void)
37                                 (count-help-2 (+ curr 1) end)))))
38     (count-help-2 0 n)))
```

```
37 | (count-help-2 (+ curr 1)
37 | end))))))
38 | (count-help-2 0 n)))
39 |
40 | (countup-2 4)
```