

```

1 #lang racket
2
3 (define (fizzbuzz n j)
4   (cond ((and (= (modulo j 3) 0) (= (modulo j 5) 0)) (printf "fizzbuzz\n"))
5         ((= (modulo j 3) 0) (printf "fizz\n"))
6         ((= (modulo j 5) 0) (printf "buzz\n"))
7         (else (printf (number->string j))))
8   (if (= n j)
9       (void)
10      (fizzbuzz n (+ j 1))))
11
12 (fizzbuzz 10 0)
13
14
15 ;; Anonymous function that returns second item from a list
16
17 ((lambda (lst) (first (rest lst))) (list 1 2 3))
18
19 ;; Original count-up function
20
21 (define (count-help x y)
22   (printf (number->string x))
23   (if (= x y)
24       (void)
25       (count-help (+ x 1) y)))
26
27 (define (count-up x)
28   (count-help 1 x))
29
30 (count-up 5)
31
32 ;; Count-up using letrec
33
34 (define (count-up-2 x)
35   (letrec ((count-help (lambda (x y)
36                         (printf (number->string x))
37                         (if (= x y)
38                             (void)
39                             (count-help (+ x 1) y))))))
40     (count-help 1 x)))
41
42 (count-up-2 5)
43
44
45

```