```racket
#lang racket

(define (fizzbuzz n j)
  (cond ((and (= (modulo j 3) 0) (= (modulo j 5) 0)) (printf "fizzbuzz\n"))
            ((= (modulo j 3) 0) (printf "fizz\n"))
            ((= (modulo j 5) 0) (printf "buzz\n"))
            (else (printf (number->string j))))
  (if (= n j)
      (void)
      (fizzbuzz n (+ j 1))))

(fizzbuzz 10 0)


;; Anonymous function that returns second item from a list

((lambda (lst) (first (rest lst))) (list 1 2 3))

;; Original count-up function

(define (count-help x y)
  (printf (number->string x))
  (if (= x y)
      (void)
      (count-help (+ x 1) y)))

(define (count-up x)
  (count-help 1 x))

(count-up 5)

;; Count-up using letrec

(define (count-up-2 x)
  (letrec ((count-help (lambda (x y)
                         (printf (number->string x))
                         (if (= x y)
                             (void)
                             (count-help (+ x 1) y)))))
    (count-help 1 x)))

(count-up-2 5)



```