```
#lang racket

(define (add-5 lst)
  (if (equal? lst null)
      null
      (cons (+ (first lst) 5)
            (add-5 (rest lst))))
  )

(add-5 (list 1 2 3))

(define (countdown n)
  (println n)
  (if (= n 0)
      (void)
      (countdown (- n 1))))

(countdown 10)

;; first-pass count-up

(define (countup n end)
  (println n)
  (if (= n end)
      (void)
      (countup (+ n 1) end)))

(countup 0 10)

;; 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)

;; fizzbuzz

(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)


```