```racket
#lang racket

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

(add-5 (list 1 2 4))

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

(countdown 10)

(define (count-help curr end)
  (println curr)
  (if (= curr end)
      (void)
      (count-help (+ curr 1) end)))

(count-help 0 4)

(define (countup n)
  (count-help 0 n))

(countup 4)

(define (countup-2 n)
  (letrec ((count-help-2 (lambda (curr end)
                           (println curr)
                           (if (= curr end)
                               (void)
```

```
                        (count-help-2 (+ curr 1)
end)))))
    (count-help-2 0 n)))

(countup-2 4)
```