```
(define result
  (letrec ((even?
            (lambda (n)
              (if (= n 0)
                  #t
                  (odd? (- n 1)))))
           (odd?
            (lambda (n)
              (if (= n 0)
                  #f
                  (even? (- n 1))))))
    (even? 5)))

(display result) ; Output: #f
```

Infinite lists

```
(define lazycubes
  (letrec ((next (lambda (n)(cons (* n (* n n))(delay(next (+ n 1)))))))(next 1))

(define naturals
  (letrec ((next (lambda (n) (cons n (delay (+ n 1)))))) (next 1)))
```

tail recursion

```
(define (factorial-tail n)
  (define (factorial-helper n acc)
    (if (= n 0)
        acc
        (factorial-helper (- n 1) (* acc n))))
  (factorial-helper n 1))
```