```racket
#lang racket

(define (fac n)
  (if (= 1 n)
      1
      (* n (fac (- n 1)))))

;; Tail-recursive version

(define (tail-fac n)
  (letrec ((helper (lambda (n acc)
                     (if (= 1 n)
                         acc
                         (helper (- n 1) (* n acc))))))
    (helper n 1)))

(tail-fac 4)

;; Version from last week

(define (string-reverse str)
  (letrec ((helper
         (lambda (x n)
           (if (= n 0)
               (string (string-ref x n))
               (string-append (string (string-ref x n))
                              (helper x (- n 1)))))))
    (helper str (- (string-length str) 1))))

(string-reverse "elephant")

;; Tail-recursive version

(define (tail-string-reverse str)
  (letrec ((helper
            (lambda (x n res)
              (if (= n 0)
                  (string-append res (string (string-ref x n)))
                  (helper x (- n 1) (string-append res (string (string-ref
x n))))))))
    (helper str (- (string-length str) 1) "")))

(tail-string-reverse "elephant")
```