```
#lang racket

(define (try a b)
  (if (= a 0)
      1
      b))

;;(try 0 (/ 1 0)) Errors in function arguments show
up immediately

(define numbers (list 1 2 3 4 5))

(define (trace op)
  (printf (string-append op "!\n")))

(define squares (map (lambda (x) (trace "Map") (* x
x)) (list 1 2 3)))
squares

(define (add x y)
  (trace "Addition")
  (+ x y))

(define (subtract x y)
  (trace "Subtraction")
  (- x y))

(define (multiply x y)
  (trace "Multiply")
  (* x y))

(define (fac n)
  (if (= n 1)
      1
      (multiply n (fac (subtract n 1)))))

```

```
(fac 5)

(define (tail-fac n)
  (letrec ((helper (lambda (x res)
                     (if (= x 1)
                         res
                         (helper (subtract x
1)(multiply x res))))))
  (helper n 1)))

(tail-fac 5)

(define (loop-forever)
  (loop-forever))

;;(loop-forever)

(define (three x)
  3)

(three (loop-forever))
```