```
#lang racket

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

;(try 0 (/ 1 0)) ; Error because Racket is
call-by-value

(define (trace)
  (printf "Operation!\n"))

(define squares (map (lambda (x)(trace)(* x
x)) (list 1 2 3 4)))
squares

(define (add x y)
  (printf "Addition!\n")
  (+ x y))

(define (subtract x y)
  (printf "Subtraction!\n")
  (- x y))

(define (multiply x y)
  (printf "Multiply!\n")
  (* 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 (three x)
  3)

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

(three (loop-forever))
```