```
#lang racket

;; Data format: quoted Racket programs

;; Environments

(define (fresh-env)
  (make-immutable-hash))

;; Base types

(define (valid-type? typ)
  (match typ
    ['number #t]
    ['bool #t]
    [(list '-> t1 t2)
     (and (valid-type? t1)
          (valid-type? t2))]))

;; type-checker

(define (type-check env exp)
  (match exp
    [(? number? n) 'number]
    [(? boolean? n) 'bool]
    [(list 'and e1 e2)
     (if (and (equal? (type-check env e1)
                      'bool)
              (equal? (type-check env e2)
                      'bool))
         'bool
         (error "arguments aren't booleans!"))]
    [(list '+ e1 e2)
     (if (and (equal? (type-check env e1)
                      'number)
              (equal? (type-check env e2)
                      'number))
         'number
         (error "arguments aren't numbers!"))]
    [(list 'lambda (list var ': (? valid-type? typ)) body)
     (list '-> typ (type-check (hash-set env var typ) body))]
    [(list fun arg)
     (let ([fun-typ (type-check env fun)]
           [arg-typ (type-check env arg)])
       (match fun-typ
         [(list '-> expected-arg-typ result-typ)
          (if (equal? arg-typ expected-arg-typ)
              result-typ
```

```
49              (error "argument type mismatch!"))]
50          [else (error "expected a function : ( ")]))]
51     [(? symbol? id)
52      (hash-ref env id)]
53     ))
54
55
```