```
;; values of expressions
(deftype Value
  (numV n)
  (closureV id body env))

;; eval :: Expr Env -> Value
(define (eval expr env)
  (match expr
    [(num n) (numV n)]
    [(fun id body) (closureV id body env)]
    [(app f e) (def (closureV the-arg the-body the-claus-env) (eval f env))
               (def the-ext-env (extend-env the-arg (eval e env) the-claus-env))
               (eval the-body the-ext-env)]
    [(id x) (env-lookup x env)]
    [(add l r) (num+ (eval l env) (eval r env))]
    [(if0 c t f) (if (num-zero? (eval c env)) (eval t env) (eval f env))]))
```