```
(define (valof expr env)
  (match expr
    [`,y #:when (symbol? y) (apply-env env y)]
    [`,n #:when (number? n) n]
    [`(add1 ,nexp) (add1 (valof nexp env))]
    [`(λ (,x) ,b)
     (make-closure x b env)]
    [`(,rator ,rand)
     (apply-closure (valof rator env) (valof rand env))]))

> (valof '((λ (x) x) 5) (empty-env))
5
```

"closure"

```
(Δ (apply-closure c a)
   (c a))
```

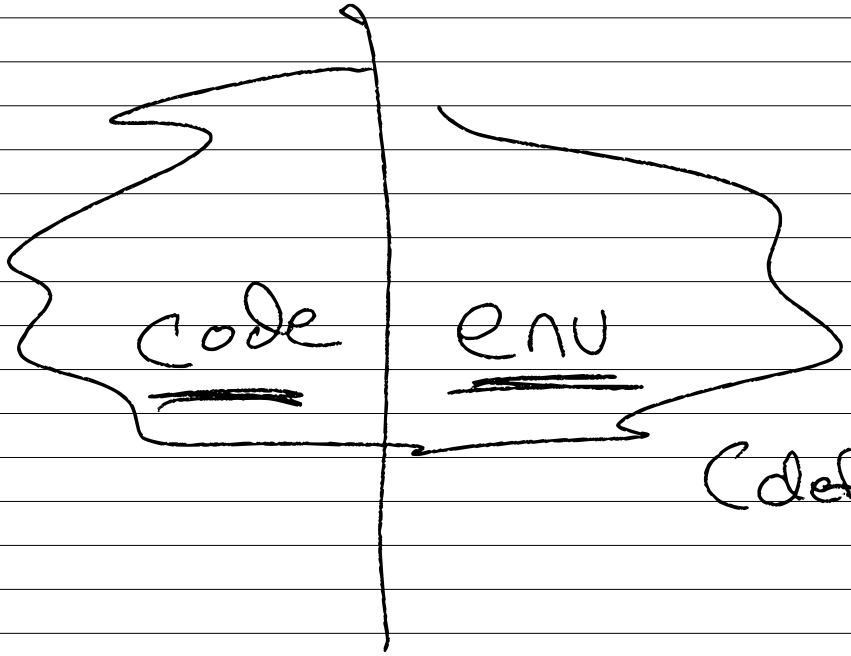
```
(Δ (make-closure x b env)
   (λ (a)
      (eval b (extend-env x a env))))
```
} closure

```
(Δ (apply-env env y)
   (env y))
```

```
(Δ (extend-env x a env)
   (λ (y) (if (eqv? x y)
              a
              (apply-env env y))))
```

← environment

```
(Δ (empty-env)
   (λ (y) 'σ))
```




```
(define f (x)
  (define g (y)
    (+ x y))
  g)
```

(g (7))

(+ x 7)