

```
;; parse :: s-expr -> Expr
;; converts s-expressions into Exprs
(define (parse s-expr)
  (match s-expr
    [ n #:when (number? n) (num n) ]
    [ x #:when (symbol? x) (id x) ]
    [(list '+ l r) (add (parse l) (parse r))]
    [(list '- l r) (sub (parse l) (parse r))]
    [(list 'if0 c t f) (if0 (parse c) (parse t) (parse f))]
    [(list 'fun (list x) b) (fun x (parse b))]
    [(list f a) (app (parse f) (parse a))]
    [(list 'with (list x e) b) #:when (symbol? x)
      (app (fun x (parse b)) (parse e))]))
```