```
;; s-expressions used as concrete
;; syntax for our arithmetical language
#|
<s-expr> ::= <num>
           | (list '+  <s-expr> <s-expr>)
           | (list '-  <s-expr> <s-expr>)
           | (list 'if0  <s-expr> <s-expr> <s-expr>)
           | (list 'with (list <sym> <s-expr>) <s-expr>)
           | <sym>
           | (list <sym> <s-expr>)
|#

;; 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 f a) #:when (symbol? f) (app f (parse a))]))
```