> (cons? '(1))

- Boolean - #t

> (cons? '())

- Boolean - #f

> (cons 1 '(1 2 3))

- (Listof Number) - '(1 1 2 3)

length : ((Listof 'a) -> Number)

Returns the number of items in a list.

//foldl or foldl with operator returns value

(foldl + 10 (list 1 2 3))

- Number – 16

(foldl cons '() '(0 1 2 3 4 5 6 7 8 9))

- (Listof Number) '(9 8 7 6 5 4 3 2 1 0)

And foldR does same thing but numbers right.

1. (run `(let ([x 7]) (+ x x)))

Trace: (list (bind x 7)) or (list (bind 'x (numV 7))), and the output (numV 14)

1. (run `(let ([x 7]) (+ 4 ((lambda (y) (\* 3 y)) x))))

Trace: (list (y 7) (x 7)) or (list (bind 'y (numV 7)) (bind 'x (numV 7)))

Or (list (bind 'x (numV 7)) (bind 'x (numV 11)) (bind 'x (numV 13))), and the output (numV 25)

1. (run `((lambda (x) (+ 3 ((lambda (x) (\* 2 ((lambda (x) (+ 5 x)) 7)))11)))13))

Trace: (list (x 7) (x 11) (x 13)), and the output (numV 27)

1. (run `{(lambda (x) x) (lambda (y) y)})

(extend-env (bind 'x (closV 'y (idE 'y) '())) '())

<(list (bind 'x (closV 'y (idE 'y) '())))

Trace: (closV 'y (idE 'y) '())

Output: (closV 'y (idE 'y) '())

Closed value expression.

(interp (parse `{lambda {x} {+ x x}})

mt-env)- Value

(closV 'x (plusE (idE 'x) (idE 'x)) '())