notes

Daniel Dyla

September 21, 2016

Contents

1 eqv? vs equal?

- equal? returns true if the elements are the same
- eqv? returns true if the elements are the same object in memory

```
(define a (range 2 5))
(define b (range 2 5))
(list (equal? a b)
      (eqv? a b))
```

2 tail recursion

2.1 Factorial

2.2 List Reversal

2.3 Zip

2.3.1 Normal Recursive

```
(define (zip l1 l2)
  (cond
   ((or (null? l1) (null? l2))
    '())
   (else
    (cons
     (list (car l1) (car l2))
     (zip (cdr l1) (cdr l2))))))

(zip '(3 4 2) '(9 2))
```

2.3.2 Tail Recursive

```
(define (zip l1 l2 acc)
  (cond
   ((or (null? l1) (null? l2))
    ; accumulator list is built in reverse
    (reverse acc))
   (else
    (zip
     (cdr l1)
     (cdr l2)
     (cons
      (list (car l1) (car l2))
      acc)))))

(zip '(3 4 2) '(9 2 9) '())
```

3 map

- map a function onto a list (map f lst)
- map a function onto a list with 1 argument per call (map f lst args)

```
(map + '(1 2 3) '(1 2 3))

; only works on lists of equal length
(define (zip l1 l2)
  (map list l1 l2))

(zip '(1 2 3) '(4 5 6))
```