Base Case Left Hand Side

L.H.S let x be '() => empty list Apply-Compare law: **Base Case**

$$((o f g) x) == (f (g x))$$

(((curry map)f)(((curry map) g) '()))

Apply-curried law: (((curry f) x) y) == (f x y)

(map f (((curry map) g) '()))

(map f(map g '()))

Apply map-empty law in the inner map (map g '()) is '()

Map-empty law in outer map (map f '())

Apply definition of map
If #t => empty law

(if (null? '()) '() => #t

if #f => Non-empty law

(Map f '())) will return '()

Inductive Case

Assume x = (cons z zs)

Compose law

(((curry map) f) (((curry map) g) x))

Curried law (map f (map g x))

X is not an empty (map f (map g (cons z zs))

Base Case Right Hand Side

Let x be '() => **Base Case**

(((curry map) (o f g)) '())

Apply curried law (map (o f g) '())

Apply definition of map If #t law

(if (null? '()) '() =>#t

if #f non-empty law

(cons (f (car '() (map f (cdr '())))) => #f

The first case is true hence: (map (o f g) '()) => '()

So both sides evaluate to '() at the base case so they are equal at their base cases.

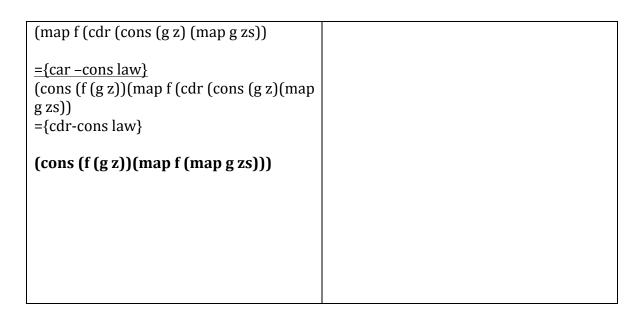
Inductive Case

Assume x = (cons z zs)

={curried-law}
(map (o f g) (cons z zs))

={Apply definition of map}

```
substitute actual parameter into inner
                                             (if (null? (cons z zs)) '()
map with definition of map
                                                 (cons ((o f g) (car (cons z zs)))
                                                   (map (o f g) (cdr (cons z zs)))
                                             =null? -cons law
(map f
  (if (mull? (cons z zs)) '()
                                             (if #f'()
  (cons (g (car (cons z zs)))
                                                (cons ((o f g) (car (cons z zs)))
     (map g (cdr (cons z zs)))))
                                                (map (o f g) (cdr (cons z zs)))
={null? - cons law}
                                             ={if -#f law}
(map f
                                             (cons (( o f g) (car (cons z zs)))
  (if #f
                                             (map (o f g) (cdr (cons z zs)))
      "()
       (cons (g (car (cons z zs)))
                                             ={car-cons law}
         (map g (cdr (cons z zs))))
                                             (cons (( o f g) z) (map (o f g)(cdr (cons z
                                             zs)))
={if #f law}
(map f
                                             ={cdr-cons law}
                                              (cons (( o f g) z) (map (o f g) zs))
  (cons (g (car (cons z zs)))
   (map g (cdr (cons z zs))))
                                             ={compose law}
={car-cons law}
                                              (cons (f (g z)) (map (o f g) zs))
(map f
  (cons (g z) (map g (cdr (cons z zs))))
                                             ={curried law}
                                             (cons (f (g z))(((curry map) (o f g)) zs)
={cdr-cons law}
(map f
                                             Using the inductive hypothesis
                                             (cons (f (g z))((o (( curry map) f) (curry map) f)))
 (cons (g z) (map g zs))
                                             map) g)) zs))
=Apply Map function Definition
                                             ={compose law}
                                             (cons (f (g z))(((curry map) f)(((curry
(if (null? (cons (g z) (map g zs)) '( )
  (cons (f (car (cons (g z) (map g zs))
                                             map) g) zs)))
     (map f (cdr (cons (g z) (map g zs))
                                             ={curried law}
={null?-cons law}
                                             (cons (f (gz))(map f g zs)))
(if #f
                                             LHS = RHS
    (cons (f car (cons (g z) (map g zs)))
    (map f (cdr (cons (g z) (map g zs))
={if -#f law}
(cons (f car (cons (g z) (map g zs)))
```



Starting equation

((o((curry map) f) ((curry map)g))x) == (((curry map) (o f g))x)