Ling Xiang
HW2
2/25/2019
Professor Zmuda

don't use term in its own definition.

15.4 - In what common data structure are Lisp lists normally stored?
    linked list ✓

15.5 - Explain why QUOTE is needed for a parameter that is a data list.
    quote means treating the parameter that is quoted as an element, instead of executing it. For example, '(+ 2 3) means this is a list of element with +, 2, 3. If we removed the quote, scheme will give us 5 instead of a list.

18

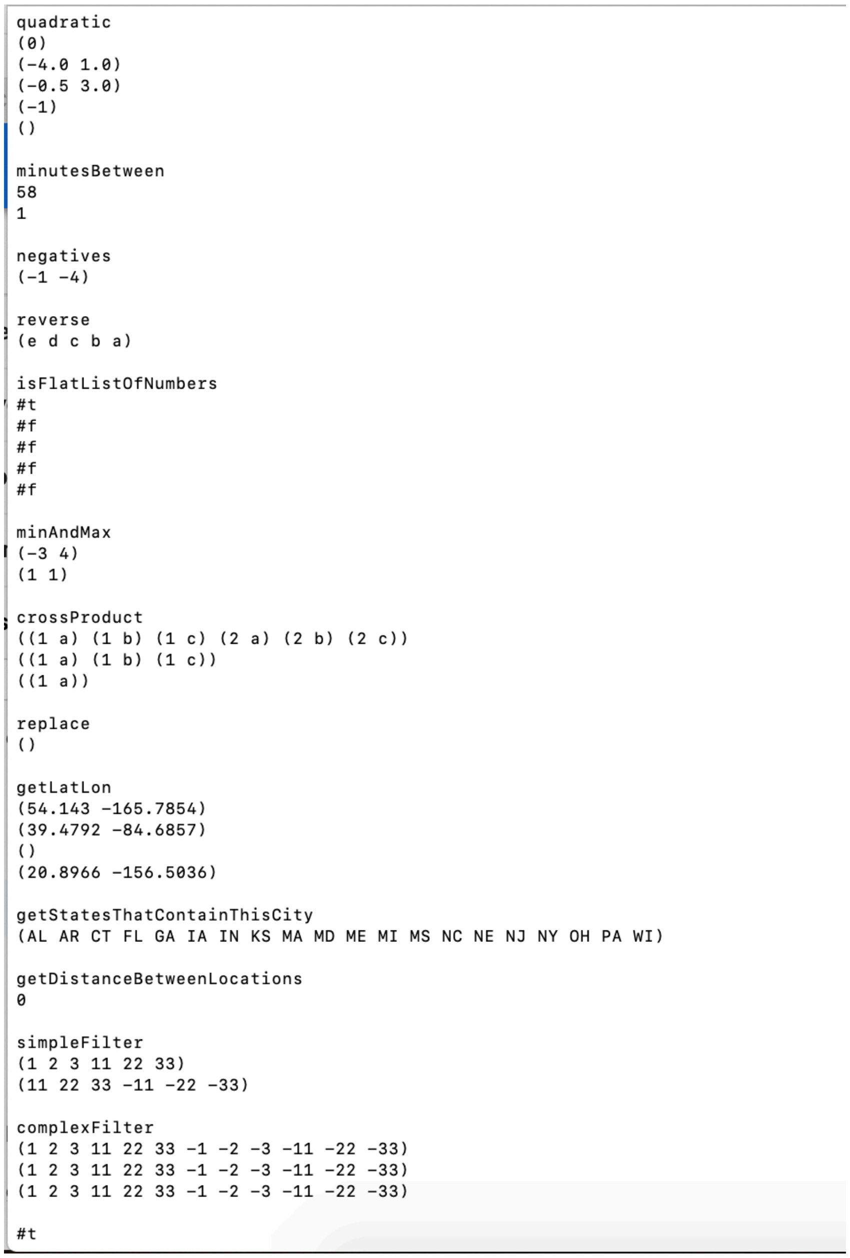
15.9 - What are the differences between =, EQ?, EQV?, and EQUAL?
    = is used in numerical element, a equivalence predicate ✓
    EQ? is to check if two elements are in the same memory address. In other words, are two parameters are the same object. ✓
    EQV? return true if two objects are normally regarded as the same object. ✓
        eg, obj1 and obj2 are both #t or both #f, EQV? returns true.
    EQUAL? is to check if two atoms are equivalent. It can be applied onto lists, vectors, strings, etc.

15.14 - If CONS is called with two atoms, say 'A and 'B, what is the returned?
    It returns (a . b)#{Unspecific} ✓
    Since both of them are atoms, it returns a dot in the middle

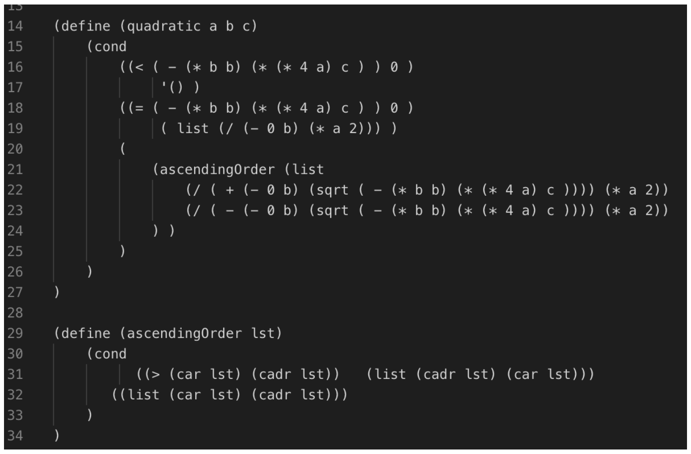
15.16 - What are the differences between CONS, LIST, and APPEND?
    CONS: add a atom to the front of a list
    LIST: ~~making two atoms~~ a list -2
    APPEND: joins two lists together to make one

Description:
    Does not work: replace, getDistanceBetweenZipCodes, complexFilter
    Others work fine.

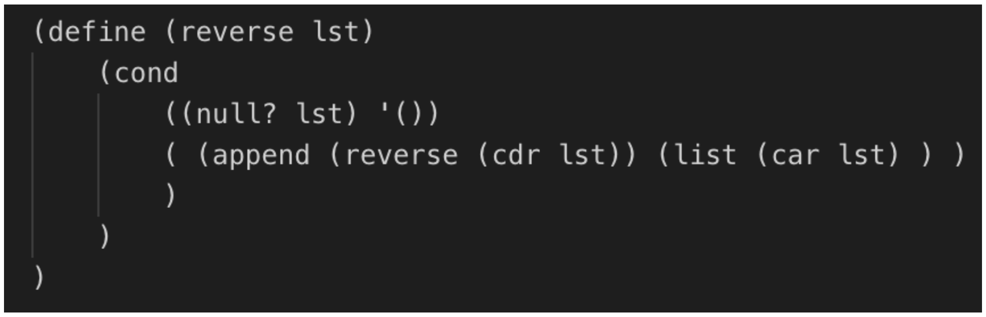
Output:



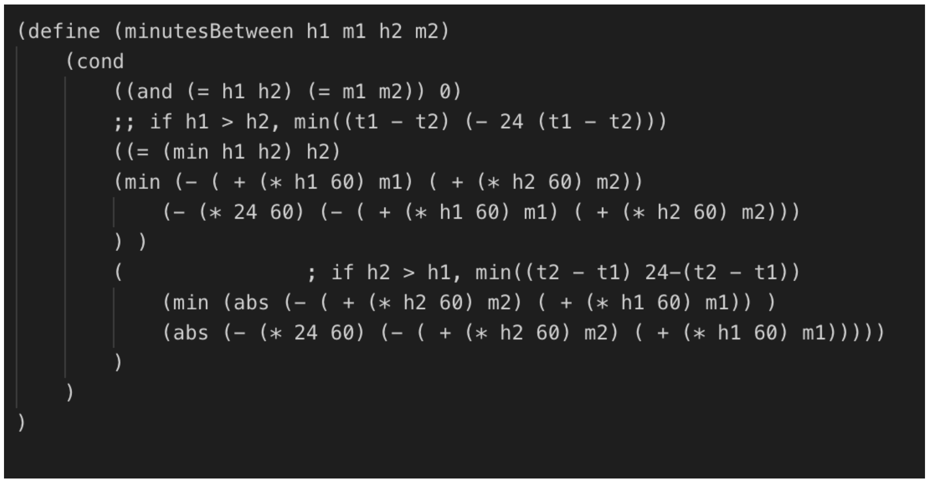
```
quadratic
(0)
(-4.0 1.0)
(-0.5 3.0)
(-1)
()

minutesBetween
58
1

negatives
(-1 -4)

reverse
(e d c b a)

isFlatListOfNumbers
#t
#f
#f
#f
#f

minAndMax
(-3 4)
(1 1)

crossProduct
((1 a) (1 b) (1 c) (2 a) (2 b) (2 c))
((1 a) (1 b) (1 c))
((1 a))

replace
()

getLatLon
(54.143 -165.7854)
(39.4792 -84.6857)
()
(20.8966 -156.5036)

getStatesThatContainThisCity
(AL AR CT FL GA IA IN KS MA MD ME MI MS NC NE NJ NY OH PA WI)

getDistanceBetweenLocations
0

simpleFilter
(1 2 3 11 22 33)
(11 22 33 -11 -22 -33)

complexFilter
(1 2 3 11 22 33 -1 -2 -3 -11 -22 -33)
(1 2 3 11 22 33 -1 -2 -3 -11 -22 -33)
(1 2 3 11 22 33 -1 -2 -3 -11 -22 -33)

#t
```



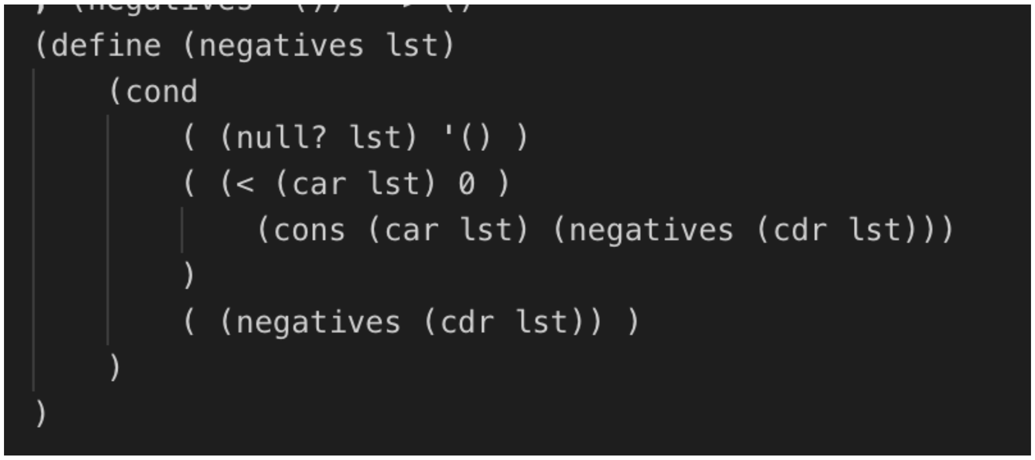
```
(define (quadratic a b c)
    (cond
        ((< ( - (* b b) (* (* 4 a) c ) ) 0 )
            '() )
        ((= ( - (* b b) (* (* 4 a) c ) ) 0 )
            ( list (/ (- 0 b) (* a 2))) )
        (
            (ascendingOrder (list
                (/ ( + (- 0 b) (sqrt ( - (* b b) (* (* 4 a) c )))) (* a 2))
                (/ ( - (- 0 b) (sqrt ( - (* b b) (* (* 4 a) c )))) (* a 2))
            ) )
        )
    )
)

(define (ascendingOrder lst)
    (cond
          ((> (car lst) (cadr lst))   (list (cadr lst) (car lst)))
       ((list (car lst) (cadr lst)))
    )
)
```



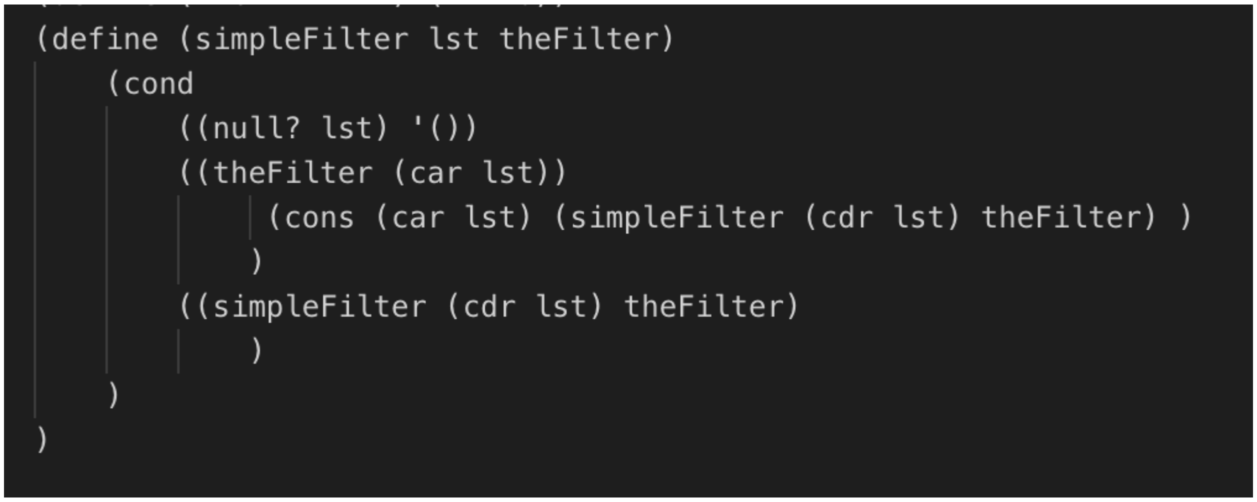
```scheme
(define (reverse lst)
    (cond
        ((null? lst) '())
        ( (append (reverse (cdr lst)) (list (car lst) ) )
        )
    )
)
```



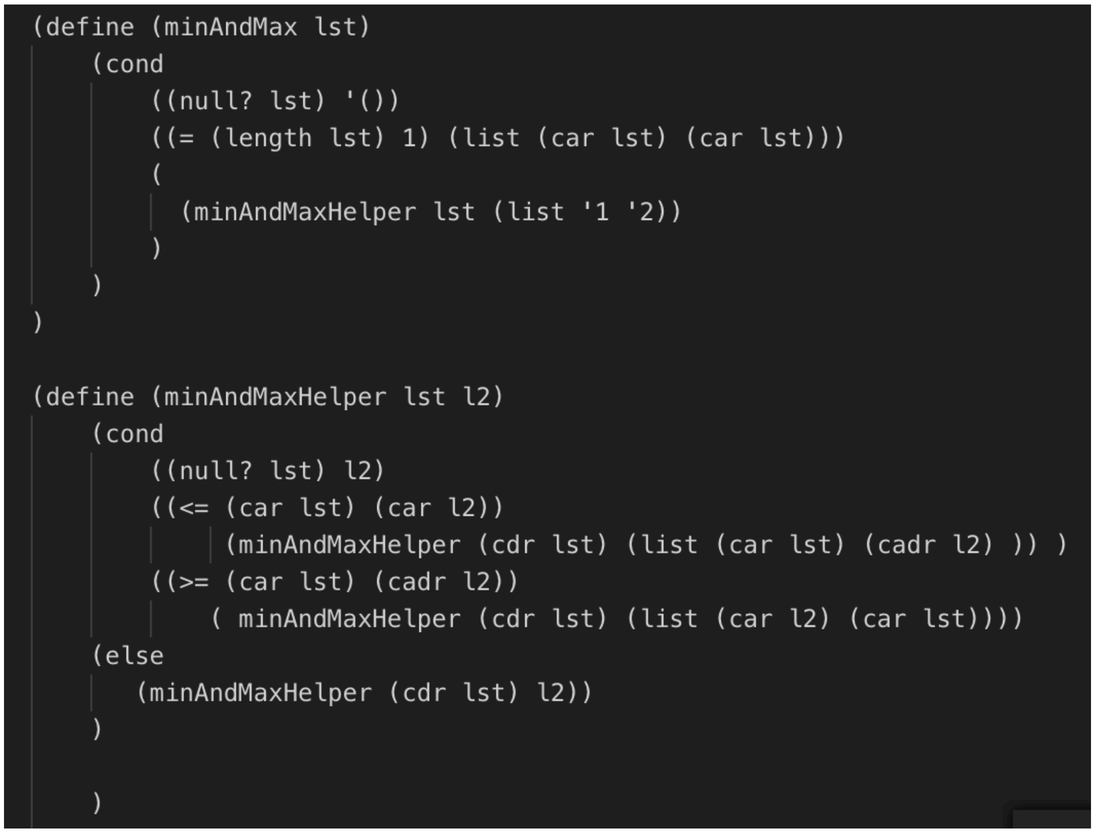
```scheme
(define (minutesBetween h1 m1 h2 m2)
    (cond
        ((and (= h1 h2) (= m1 m2)) 0)
        ;; if h1 > h2, min((t1 - t2) (- 24 (t1 - t2)))
        ((= (min h1 h2) h2)
        (min (- ( + (* h1 60) m1) ( + (* h2 60) m2))
            (- (* 24 60) (- ( + (* h1 60) m1) ( + (* h2 60) m2)))
        ) )
        (               ; if h2 > h1, min((t2 - t1) 24-(t2 - t1))
            (min (abs (- ( + (* h2 60) m2) ( + (* h1 60) m1)) )
            (abs (- (* 24 60) (- ( + (* h2 60) m2) ( + (* h1 60) m1)))))
        )
    )
)
```



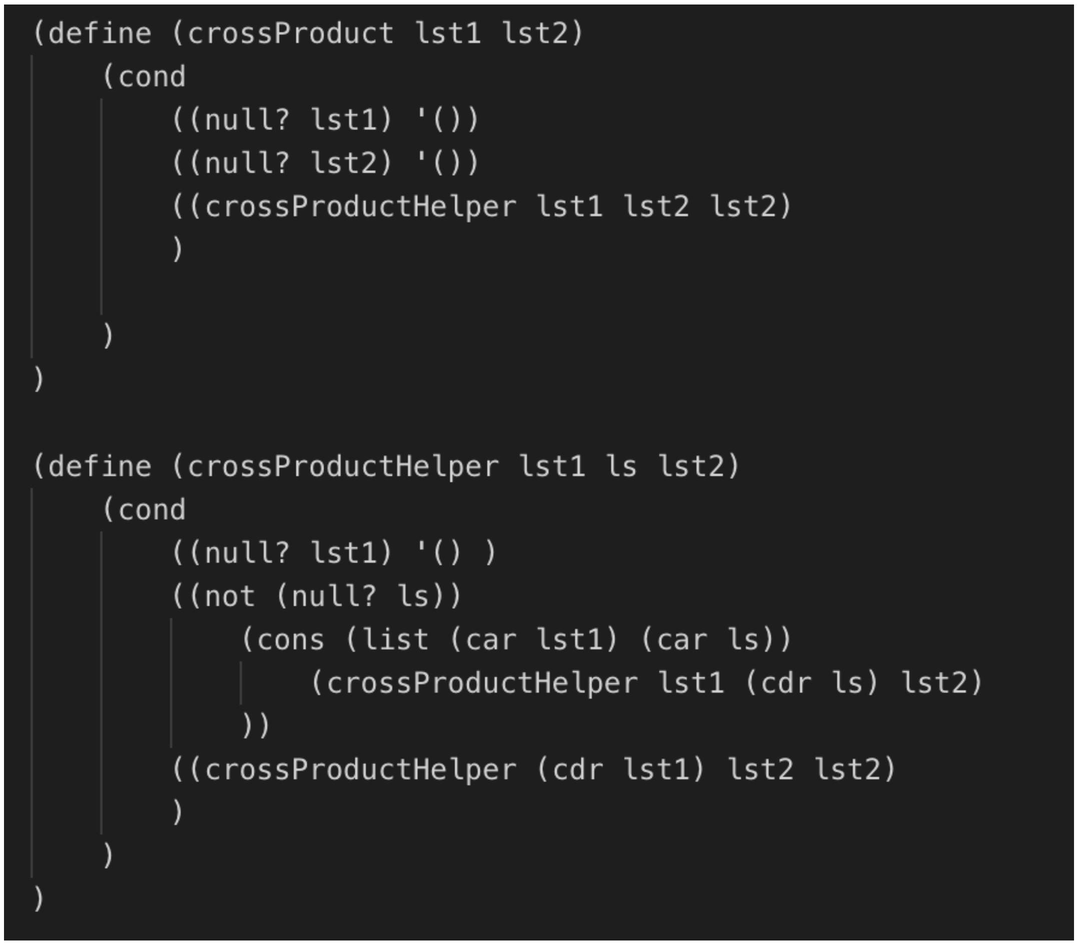
```scheme
(define (negatives lst)
    (cond
        ( (null? lst) '() )
        ( (< (car lst) 0 )
            (cons (car lst) (negatives (cdr lst)))
        )
        ( (negatives (cdr lst)) )
    )
)
```



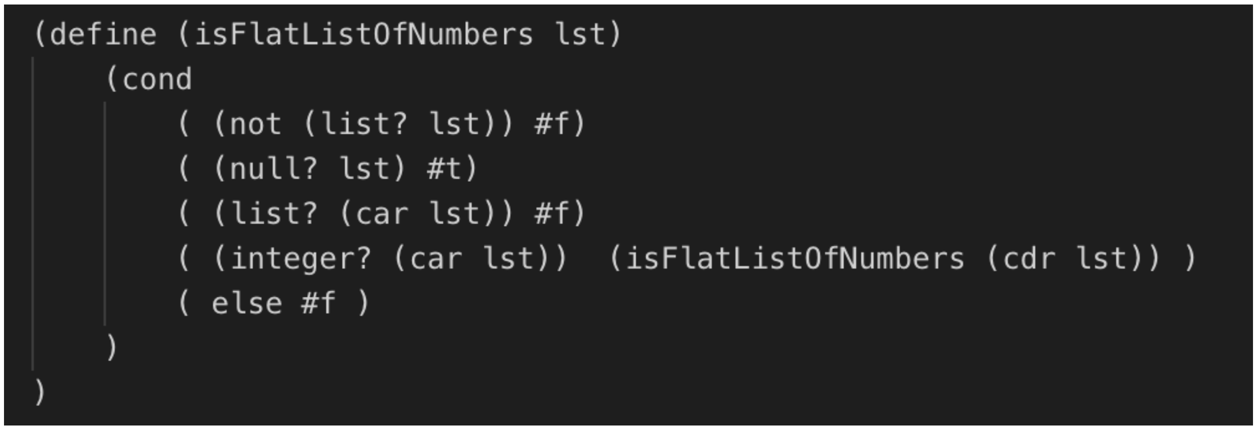
```scheme
(define (simpleFilter lst theFilter)
    (cond
        ((null? lst) '())
        ((theFilter (car lst))
                (cons (car lst) (simpleFilter (cdr lst) theFilter) )
                )
        ((simpleFilter (cdr lst) theFilter)
                )
    )
)
```



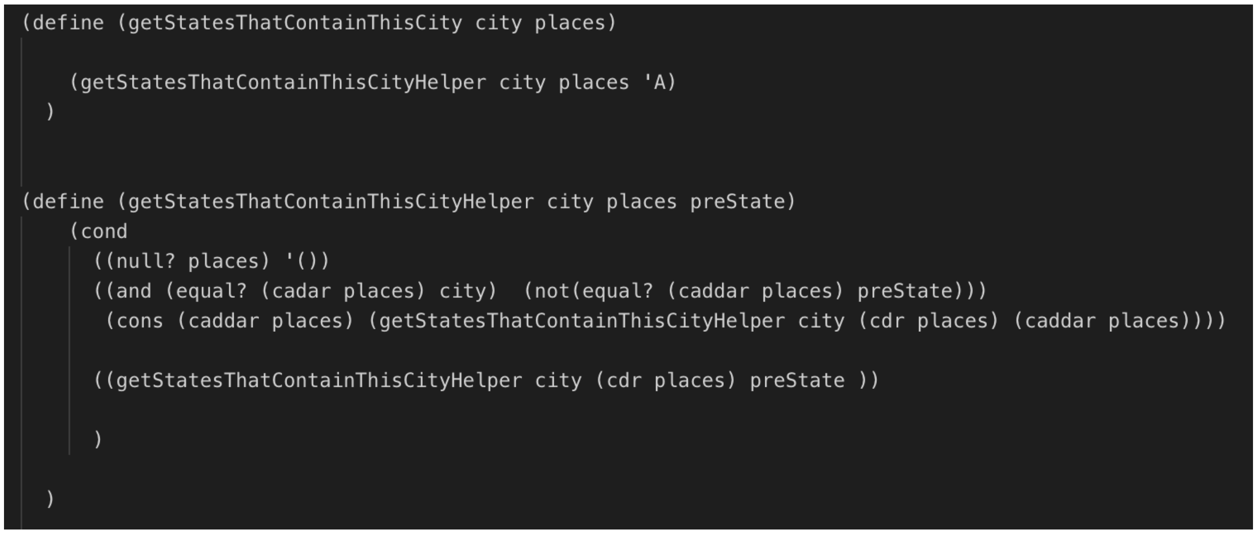
```
(define (minAndMax lst)
    (cond
        ((null? lst) '())
        ((= (length lst) 1) (list (car lst) (car lst)))
        (
          (minAndMaxHelper lst (list '1 '2))
        )
    )
)

(define (minAndMaxHelper lst l2)
    (cond
        ((null? lst) l2)
        ((<= (car lst) (car l2))
             (minAndMaxHelper (cdr lst) (list (car lst) (cadr l2) )) )
        ((>= (car lst) (cadr l2))
            ( minAndMaxHelper (cdr lst) (list (car l2) (car lst))))
    (else
       (minAndMaxHelper (cdr lst) l2))
    )

    )
```



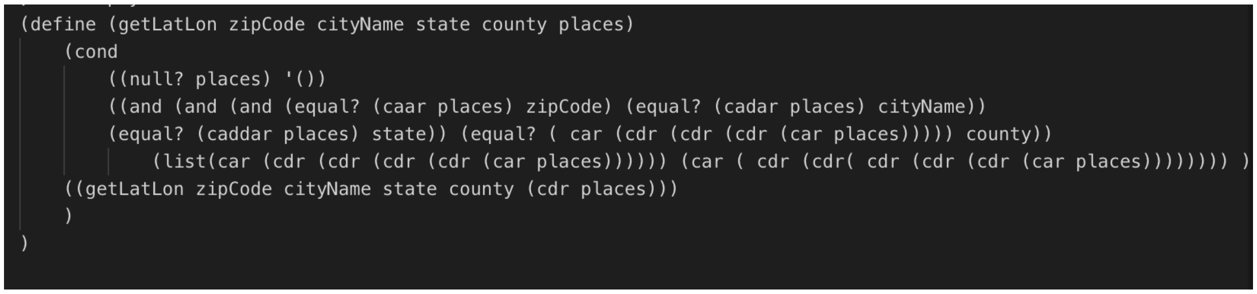
```
(define (crossProduct lst1 lst2)
    (cond
        ((null? lst1) '())
        ((null? lst2) '())
        ((crossProductHelper lst1 lst2 lst2)
        )

    )
)

(define (crossProductHelper lst1 ls lst2)
    (cond
        ((null? lst1) '() )
        ((not (null? ls))
            (cons (list (car lst1) (car ls))
                (crossProductHelper lst1 (cdr ls) lst2)
            ))
        ((crossProductHelper (cdr lst1) lst2 lst2)
        )
    )
)
```



```
(define (isFlatListOfNumbers lst)
    (cond
        ( (not (list? lst)) #f)
        ( (null? lst) #t)
        ( (list? (car lst)) #f)
        ( (integer? (car lst))  (isFlatListOfNumbers (cdr lst)) )
        ( else #f )
    )
)
```



```
(define (getStatesThatContainThisCity city places)

    (getStatesThatContainThisCityHelper city places 'A)
  )



(define (getStatesThatContainThisCityHelper city places preState)
    (cond
      ((null? places) '())
      ((and (equal? (cadar places) city)  (not(equal? (caddar places) preState)))
       (cons (caddar places) (getStatesThatContainThisCityHelper city (cdr places) (caddar places))))

      ((getStatesThatContainThisCityHelper city (cdr places) preState ))

      )

  )
```



```
(define (getLatLon zipCode cityName state county places)
    (cond
        ((null? places) '())
        ((and (and (and (equal? (caar places) zipCode) (equal? (cadar places) cityName))
        (equal? (caddar places) state)) (equal? ( car (cdr (cdr (cdr (car places))))) county))
            (list(car (cdr (cdr (cdr (cdr (car places)))))) (car ( cdr (cdr( cdr (cdr (cdr (car places)))))))) )
    ((getLatLon zipCode cityName state county (cdr places)))
    )
)
```