Concepts of Programming Languages, CSCI 305, Fall 2021
Lab 1, Scheme, Aug. 27

1. Define the *sum* function that takes two numbers and returns the sum of the numbers.

```
(define sum
        (lambda (n1 n2) (+ n1 n2))
)
```

2. Define the *sumList* function that takes a list of numbers and returns the sum of them. If the list is empty, return 0.

```
(define sumList
        (lambda (l)
                (if (null? l)
                        0
                        (+ (car l) (sumList (cdr l)))
                )
        )
)

Alternate definition that uses the 'sum' function.
(define sumList
        (lambda (l)
                (if (null? l)
                        0
                        (sum (car l) (sumList (cdr l)))
                )
        )
)
```

3. Define the function *average* that takes two numbers and returns the average of the numbers.

```
(define average
        (lambda (x y)
                (/ (+ x y) 2)
        )
)
```

4. Define the function *averageList* that takes a non-empty list of numbers and returns the average of the numbers in the list.

```
(define averageList
        (lambda (list)
                (/ (sumList) (length list))

)
```

5. Define the function *myLast* that acts like car but returns the last element of a list.

```
(define myLast
        (lambda (l)
                (if     (null? (cdr l))
                        (car l)
                        (myLast (cdr l))
                )
        )
)

or

(define myLast
        (lambda (l)
                (car (reverse  l))
        )
)
```

6. Define a function *convertFC* that takes a Fahrenheit measurement an converts it to a Celsius measurement where:

Celsius = (Fahrenheit – 32) 5/9

```
(define convertFC
        (lambda (f)
                (/ (* (- f 32) 5) 9)
        )
)
```

7. Define a function *convertFCList* that takes a list of Fahrenheit measurements and converts them to a list of Celsius measurements where:

Celsius = (Fahrenheit – 32) 5/9

```
(define convertFCList
        (lambda (l)
                (if (null? l)
                        '()
                        (cons (convertFC (car l)) (convertFCList (cdr l)))
                )
        )
)

or

(define convertFCList
        (lambda (l)
                (map converFC l)
        )
)
```

8. Define the function *inRange?* that takes a low value, a high value and a number and returns true, #t, if the number falls within the low to high range (inclusively); and false otherwise.

```
(define inRange?
        (lambda (low high item)
                (if  (or  (< item low)
                          (> item high)
                     )
                 #f
                #t
                )
        )
)
```

9. Define the function filterL*ist* that takes a low value, a high value and a list and produces a list of the values in the original list which fall within the low to high range (inclusively).

```
(define filterList
        (lambda (low high l)
                (cond
                        ((empty? l) '())
                        ((inRange? low high (car l))  (cons (car l)
                                                            (filterList low high (cdr l)))
                        (else (filterList low high (cdr l))))
                )
        )
)
```

10. Define a function *eliminateExpList* (for eliminate expensive) which takes a maximum price and a list of prices. It returns the list of prices with those prices that exceed the maximum removed. (You can assume all prices are positive.)

```
(define eliminateExpList
        (lambda (max l)
                (filterList 0 max l)
        )
)
```