Matthew Koken
COEN 171
Quiz 7

1. Explain why QUOTE is needed for a parameter that is a data list. In LISP

In lisp, by default, everything is evaluated. An expression like (+ a 2) would first lookup the value for a and then proceed to evaluate (+ <value of a> 2). The quote (') stops this evaluation. In the case that you do want to evaluate arguments, quote is needed here too. If you are evaluating a list of strings, for instance, you don't want the returned string values to be evaluated as a variable, so you would need to use QUOTE to prevent it's evaluation. Here, we can selectively evaluate the arguments from the data.

2. What is an anonymous function and when are they useful. write an example in Scala

Anonymous functions, or function literals, are instantiated as objects at runtime – as function values. The functions do not have names, hence being anonymous. These anonymous functions allow for the definition of an operation at runtime on the specific variables. Since anonymous functions are instantiated as objects, you can also pass them and save them into a val.

```
val pow2 = (x: Int) => x*x

val four = pow2(2)

{ x: Int => //also written this way

    x*x

}
```

3. What support does Python provide for functional programming

Python includes some functional like features, though the intent is not to create a functional language. It is a multi-paradigm language, so some elements can be found and used. It has Lambdas that contain an expression. Python does not optimize tail-call operations like tail-recursion, so this hampers performance. Function composition, partial application, infix operators, higher order functions are all supported, though the syntax becomes more verbose. Immutability is not supported, which conflicts with pure functional languages support and use of immutability. Overall, Python permits writing code in the style of functional programming, but as a multiple-paradigm language, it is not as clean or optimized as a language built with the intent of functional programming.

4. Why can concurrency be easier with functional programming

Functional languages don't typically rely on mutations of a variable, instead using immutable values more often. Because of this, shared states aren't necessary which removes a common issue with concurrent programming – attempting to access the same variables. This mostly avoids race conditions that can break an entire program. Functional languages also support many parallel paradigms. The focus is also typically more on high-level operations, which consist of many low level operations that are easily parallelized.

5. Write a function using Sheme syntax that takes a simple list of numbers as a parameter and returns a list with the largest and smaller number in the input list

```
(define (max arr)
  (cond ((null? (cdr arr)) (car arr))
    (( (car arr) (min (cdr arr))) (car arr))
    else (min (cdr arr))))
)


(define (min arr)
  (cond ((null? (cdr arr)) (car arr))
    ((< (car arr) (min (cdr arr))) (car arr))
    else (min (cdr arr))))
)
(define (minMax arr) (list (max arr) (min arr))
```

6. Define currying.

Currying breaks down a function taking multiple arguments into an evaluation of a series of functions that take a single argument. It reduces the complex function into smaller, more manageable parts that return functions to take another argument.

7. What is an immutable variable?

An immutable variable is like a constant – it's own value cannot be changed. The value is bound at assignment and cannot be modified or mutated.

8.Write a function in Scheme that has two input list and returns a list that is the combination of the above list with no repeated values (the input list contain only atomics values -no sublists)

```
(define (merge first second)
    (cond ((null? first)
      second)
      ((null? second)
        first)
      (else  (cons (car first) (cons (car second) (merge (cdr first) (cdr second)))))
)
```