Continuations

The Ultimate Control Structure
Suppose expression E contains a subexpression S

The continuation of S in E consists of all of the steps needed to complete E after the completion of S.

At any point during a computation the current continuation is the continuation of whatever expression is currently executing. Note that the current continuation is constantly changing.
For example,

\[(\text{define fact (lambda (n) (if (= 0 n) 1 (* n (fact (- n 1))))))}\]

Consider expression E:  \(\text{(printf "5! = ~s" (fact 5))}\)

The continuation in E of (fact 5) is the call to printf.

The continuation of (fact 4) is "multiply the result by 5, then call printf".

The continuation of (fact 3) is "multiply the result by 4, multiply the result of that by 5, then call printf."

Note that the continuations become increasingly complex as we proceed through the recursion.
Now consider fact-acc:

\[
\text{(define fact-acc (lambda (n acc)
    (if (= 0 n) acc
        (fact-acc (- n 1) (* n acc)))))}
\]

Let E be the expression (printf "5! = \~s" (fact-acc 5 1))

The continuation of (fact-acc 5 1) is the printf statement.

The continuation of (fact-acc 4 5) is the printf statement.

The continuation of (fact-acc 3 20) is the printf statement.

and so forth.

Note that the continuation doesn't change as we go back through the recursion.
The continuation of a deep recursion becomes more complex as the recursion progresses. The continuation of a tail recursion remains constant as the recursion progresses.