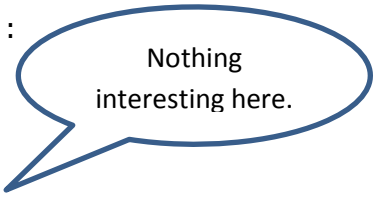
Let's have a look at some of the continuations when Scheme processes the following expression.

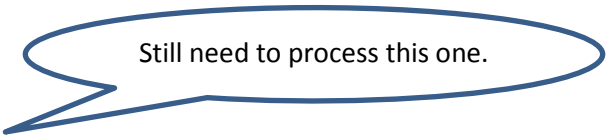
```
(+ 3 (call/cc (lambda (k) (+ 4 (k 5) 6))) 7)
```

Output of parser:

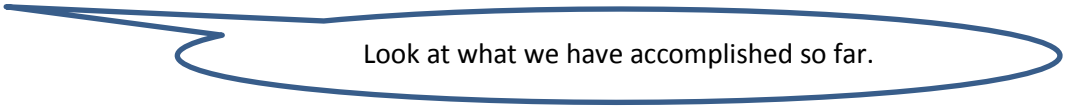



```
(app-exp
  ((var-exp +)
    (lit-exp 3)
    (callcc-exp
      (lambda-exp
        (k)
        (app-exp
          ((var-exp +)
            (lit-exp 4)
            (app-exp ((var-exp k) (lit-exp 5)))
            (lit-exp 6)))))
    (lit-exp 7)))
```

The continuation when we evaluate the call/cc-exp:

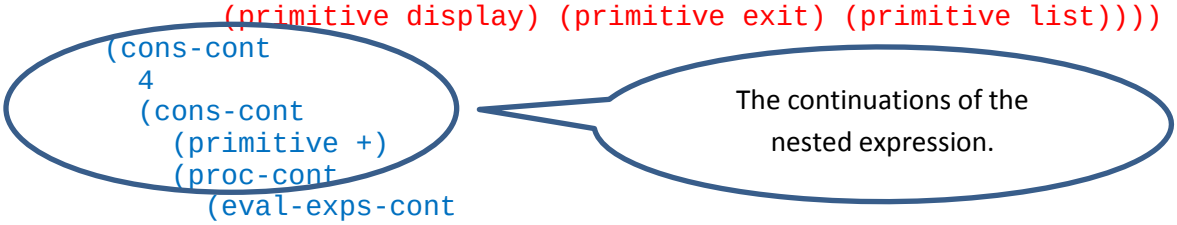



```
(eval-exps-cont
  ((lit-exp 7))
  (((+ - * / display exit list)
     .
     #7((primitive +) (primitive -) (primitive *) (primitive /)
        (primitive display) (primitive exit) (primitive list))))
  (cons-cont
    3
    (cons-cont (primitive +) (proc-cont (halt-cont)))))
```



The continuation right before we evaluate (k 5). The eval-exps-cont has the remaining expressions, the environment and the continuation at the time:

```
(eval-exps-cont
  ((lit-exp 6))
  (((k) .
        #1((acontinuation
             (eval-exps-cont
               ((lit-exp 7))
               (((+ - * / display exit list)
                  .
                  #7((primitive +) (primitive -) (primitive *)
                     (primitive /) (primitive display) (primitive exit)
                     (primitive list))))
               (cons-cont
                 3
                 (cons-cont (primitive +) (proc-cont (halt-cont))))))))
    ((+ - * / display exit list)
      .
      #7((primitive +) (primitive -) (primitive *) (primitive /)
        (primitive display) (primitive exit) (primitive list))))
    (cons-cont
      4
      (cons-cont
        (primitive +)
        (proc-cont
          (eval-exps-cont
            ((lit-exp 7))
            (((+ - * / display exit list)
               .
               #7((primitive +) (primitive -) (primitive *) (primitive /)
                  (primitive display) (primitive exit) (primitive list))))
            (cons-cont
              3
              (cons-cont (primitive +) (proc-cont (halt-cont))))))))))
```



In apply-proc, we will call apply-cont with the continuation k (see below) and 5.

```
 (eval-exps-cont
   ((lit-exp 7))
   (((+ - * / display exit list)
      .
      #7((primitive +) (primitive -) (primitive *) (primitive /)
         (primitive display) (primitive exit) (primitive list))))
   (cons-cont
     3
     (cons-cont (primitive +) (proc-cont (halt-cont)))))
```

The eval-exps-cont will lead to a call of eval-expression with (lit-exp 7) and another cons-cont:

```
(cons-cont
  5
  (cons-cont
    3
    (cons-cont (primitive +) (proc-cont (halt-cont)))))
```

This will eventually evaluate to 15.