CSSE 304 Day 29Summary `call/cc`:

1. **call/cc definition**
   a. **call/cc** is an abbreviation for **call-with-current-continuation**
   b. call/cc is a procedure that takes one argument; the argument is a *receiver*.
   c. This receiver is a procedure that takes one argument; that argument (in this case) is a *continuation*.
   d. A continuation is a procedure (that takes one argument); that continuation embodies the context of the application of **call/cc**. It is an escape procedure.
   e. The application **(call/cc receiver)** has the same effect as **(receiver continuation)**, where the continuation is
   f. an escape procedure that embodies the execution context of the entire **call/cc** expression.
2. **call/cc definition summary**:
   a. **(call/cc receiver)** ➔ (**receiver continuation)**, Hence the name: call-with-current-continuation.
   b. **Rephrasing it:** What is that continuation?
      If **c** is a procedure that represents the execution context of this application of **call/cc**, then the continuation is equivalent to **(escaper c).**
3. **call/cc example**
   a. **(call/cc receiver) ➔(receiver continuation)**,
   b. Consider `(+ 3 (call/cc (lambda (k) (* 2 (k 5)))))`
   c. The receiver r1 is (the procedure created by evaluating) `(lambda (k) (* 2 (k 5)))`
   d. The context c1 is(the procedure created by evaluating) `(lambda (v) (+ 3 v))`
   e. The continuation k1 is (escaper c1)
   f. Thus `(+ 3 (call/cc (lambda (k) (* 2 (k 5)))))` is equivalent to
      `(+ 3 (call/cc r1))`
      ➔ `(+ 3 (r1 k1))` definition of call/cc
      ➔ `(+ 3 (* 2 (k1 5)))` normal Scheme application of a procedure.
      ➔ `(k1 5)` k1 is an escape procedure.
      ➔ `8` Normal Scheme evaluation.

4. More `call/cc` examples

```
   (+ 3 (call/cc (lambda (k) (* 2 5))))
```

r2: `(lambda (k) (* 2 5))`
k2: same as k1.
`(+ 3 (call/cc r2))` ➔ `(+ 3 (r2 k2))` ➔ `(+ 3 10)` ➔ `13.` Escape procedure **k2** is never called.

```
(+ 3 (call/cc (lambda (k)   (k (* 2 5)))))
```

5.
```
(define xxx #f)
(+ 5 (call/cc (lambda (k)
                (set! xxx k)
                2))) ; xxx is equivalent to?
 (* 7 (xxx 4))
```

6. `(call/cc procedure?)`

7. `list-index` example is detailed in the slides:

8. **((car (call/cc list)) (list cdr 1 2 3))**

```
(let ([f 0] [i 0])
  (call/cc (lambda (k) (set! f k)))
  (printf "~a~n" i)
  (set! i (+ i 1))
  (if (< i 10) (f "ignore")))
```

9.
```
(define strange1
  (lambda (x)
    (display 1)
    (call/cc x)
    (display 2)))

(strange1 (call/cc (lambda (k) k)))
```

```
(define strange2  ; try this one yourself soon
 (lambda (x)
   (display 1)
   (call/cc (lambda (j) (x j)))
   (display 2)
   (call/cc (lambda (c) (x c)))
   (display 3)))

(strange2 (call/cc (lambda (k) k)))
```