CONTINUATIONS\
\
General form: (call/cc f) creates a continuation for that statement and applies f to it.\
So f is some function that wants one argument (say its formal parameter name in f is c).\
When (c x) is executed in f, the original (call/cc f) evaluates to x and control continues from that statement.\
\
Specific form: (the only kind in my tutorial examples so far---bad idea); We write "our own f", a one-parameter function that "captures the continuation", gives it a name we can use, as demonstrated in the lectures. That is, (call/cc (lambda (k) \'85..)), where our home-grown function can call k itself, put it in some global variable, etc.\
\
Now hold my beer and watch this...\
\
Two functions A, B: sub-Numbers etc refer to events in sequence listed below.\
\

```
\f1 define (A whereB)                    define(B whereA)\
           2                                4   ^\
                                                X \
                                           . . .\
 . . .                                (whereA  ret-val)
\
                                        5  \
(set! whereB (call/cc whereB)     )\
 7           3             6  V\
                             X\
```

;; the call\
(A B)\
 1\
--------\
Note: triple pun in A: whereB is its formal parameter, thus a\
local variable changed by set!, and a function to be applied to\
the continuation created by call/cc.\
\
1. A is called with arg B.\
2. whereB <-B by binding.\
3. later in A, continuation created, passed to whereB, which is B.\
4. the call to B binds the continuation to whereA.\
5. later in B, whereA is called with argument ret-val.\
6. the continuation whereA takes us to point X in A, with the\
   call/cc form evaluating to ret-val.\
7. finally, whereB is set! to ret-val.\
\
\
\
 Now if B uses a statement exactly like A's call/cc to call whereA, then you can see A and B are handing off control to each other, each passing its "start me here when you're ready" continuation to the other. See final exam 2013 for runnable code example.

\
}