1. This prints

The continuation `xyz` is bound to an anonymous procedure which
takes one argument (in this case, it will be 11). It ignores its
argument and continues with the execution of the `(begin...)` at
`(display 6).

The final `()` that is printed is the "value" of the final `(display
6)`, which is printed by the read-eval-print loop of the Scheme
interpreter. Note that the read-eval-print loop prints its output
on a separate line.

```
(begin
  (display 3)
  (call/cc
    (lambda (xyz)
      (display 4)
      (xyz 11)
      (display 5)))
  (display 6))
```

The continuation that is passed as `xyz` is

```
(lambda (val)
  (print val)
  (exit))
```

Well, sort of. The "print" really is the print of the top-level
read-eval-print loop. As such it might differ slightly from one
implementation to another. For instance in UMB Scheme, it inserts a
preliminary newline. That's why the 11 is printed on its own line. In
Dr. Scheme, on the other hand, no preliminary newline is inserted by the
read-eval-print loop, and so the output of this is simply 3411. Also, in
Dr. Scheme using the R5RS language, `call/cc` is not defined as a synonym for
`call-with-current-continuation`, so you have to either type out
`call-with-current-continuation` or
```
(define call/cc call-with-current-continuation).
```

2. This prints

The continuation `xyz` is bound to the print part of the read-eval-print loop
of the Scheme interpreter. Therefore, the 11 is (in UMB Scheme; see below)
printed on a separate line by the interpreter and nothing further is
printed.

```
(begin
  (display 3)
  (call/cc
    (lambda (xyz)
      (display 4)
      (xyz 11)
      (display 5)))
  )
```

The continuation that is passed as `xyz` is

```
(lambda (val)
  (display val)
  (exit))
```

3. This prints

The continuation `xyz` is now bound to the display that accepts the
value of the `call/cc`. As in the first example, the final `()` is
the "value" of the final `(display 6).

```
(begin
  (display 3)
  (display (call/cc
    (lambda (xyz)
      (display 4)
      (xyz 11)
      (display 5)))
  )
  (display 6))
```

The continuation that is passed as `xyz` is