Excercise 1.5

Josh H

January 3, 2024

He defines the following two procedures:

```
(define (p) (p))
(define (test x y)
  (if (= x 0) 0 y))
```

Then he evaluates the expression

```
(test 0 (p))
```

Consider the interpreter that uses applicative-order evaluation. The interpreter does the following:

```
(test 0 (p))
(test 0 (p))
.
.
.
```

The body of the procedure `test` never has its formal parameters substituted for the arguments because the argument `(p)` always evaluates to itself.

Consider now the interpreter that uses normal-order evaluation. The interpreter does the following:

```
(test 0 (p))
(if (= 0 0) 0 (p))
(if #t 0 (p))
0
```

The body of the procedure `test` has its formal parameters replaced by the operands. The body contains the special form `if` so the predicate expression is evaluated first. It results to being true, and we, at this point, assume that the value of the predicate expression replaces the predicate itself. The consequent is then evaluated and the whole expression reduces to `0`.