Exercise 1.5.

Ben Bitdiddle has invented a test to determine whether the interpreter he is faced with is using applicative-order evaluation or normal-order evaluation. 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))
```

What behavior will Ben observe with an interpreter that uses applicative-order evaluation? What behavior will he observe with an interpreter that uses normal-order evaluation? Explain your answer. (Assume that the evaluation rule for the special form if is the same whether the interpreter is using normal or applicative order: The predicate expression is evaluated first, and the result determines whether to evaluate the consequent or the alternative expression.)

Solution

Normal Order

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

⬇

```
(if (= x 0)
    0
    y)
```

⬇

```
(if (= 0 0)
   0
  (p))
```

⬇ only evaluates the condition (predicate)

```
(if (true)
   0
   (p))
```

⬇ if only expands / evaluates one branch

```
0
```

Applicative order

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

applicative order first evaluates all operands

⬇

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

(p) expands to (p) and therefore the expansion enters infinite cycle

⬇

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

⬇

...