Exercise 4.25.

Suppose that (in ordinary applicative-order Scheme) we define `unless` as shown above and then define `factorial` in terms of `unless` as

```
(define (factorial n)
  (unless (= n 1)
          (* n (factorial (- n 1)))
          1))
```

What happens if we attempt to evaluate `(factorial 5)`? Will our definitions work in a normal-order language?

Answer.

The evaluator drops into an infinite recursion when we attempt to evaluate `(factorial 5)`:

```
(factorial 5)
;Aborting!: maximum recursion depth exceeded
```

We can exploit the rule of substitution to trace the behavior of the evaluator:

```
(factorial 5)

(unless (= 5 1)
        (* 5 (factorial (- 5 1)))
        1)

(unless (= 5 1)
        (* 5 (unless (= 4 1)
                     (* 4 (factorial (- 4 1)))
                     1))
        1)

...
```

In ordinary applicative-order Scheme, all the arguments must be evaluated before the compound procedure is applied. In this case the evaluator must first obtian the value of `(* n (factorial (- n 1)))` before get `unless` applied. But this expression contains another call to `factorial` and must also be expanded into `(unless ...)`. The interpreter perpetually strips off the expression `(* n (factorial (- n 1)))` and make no reduction in this process, even when the predicate `(= n 1)` holds. This led the computation into the senario of infinite recursion.

This definition will work in a normal-order language, for normal-order languages delay evaluation of procedure arguments untill the actual argument values are needed. Additionally, as `unless` is defined in terms of the special form `if`, which reduces the expression just in time. Hence, we can succssively process the computation of `(factorial 5)` to the base case and obtain the result:

```
(factorial 5)

(unless (= 5 1)
        (* 5 (factorial (- 5 1)))
        1)

(if (= 5 1)
    1
    (* 5 (factorial (- 5 1))))

(* 5 (factorial (- 5 1)))
```


Email address: informlarry@gmail.com

```
(* 5 4 3 2 (factorial 1))

(* 5 4 3 2 (unless (= 1 1)
                   (* 1 (factorial (- 1 1)))
                   1))

(* 5 4 3 2 (if (= 1 1)
               1
               (* 1 (factorial (- 1 1)))))

(* 5 4 3 2 1)

120
```