**Exercise 2.5.**

Show that we can represent pairs of nonnegative integers using only numbers and arithmetic operations if we represent the pair $a$ and $b$ as the integer that is the product $2^a\,3^b$. Give the corresponding definitions of the procedures `cons`, `car`, and `cdr`.

**Answer.**

According to the fundamental theorem of arithmetic, which states that every integer greater than 1 is either prime itself or is the product of prime numbers, and that, although the order of the primes in the second case is arbitrary, the primes themselves are not. Note that 2 and 3 here are relatively prime to each other, for each nonnegtive integer, its corresponding value of $a$ and $b$ is therefore unique in terms of formula $2^a\,3^b$. Hence, we can represent pairs of nonnegtive integers with numbers and arithmetic operations in the form of $2^a\,3^b$.

This leads to a straightforward definition of `cons`:

```
(define (cons a b)
  (* (expt 2 a)
     (expt 3 b)))
```

To extract its corresponding value of $a$ and $b$ from a given nonnegtive integer, a usual practice would be by factorization. That is, we can obtain the value of $a$ by keeping dividing the integer by 2 and counting this operation in the process. When the integer becomes indivisable, the counter reveals to be the value of $a$. And the value of $b$ can be obtained in the same way.

Hence, we can accomplish to extract the exponent on a base of a given integer with a procedure namely `extract-expt`. With the help of Table 1, which shows the evolution of process in evaluating

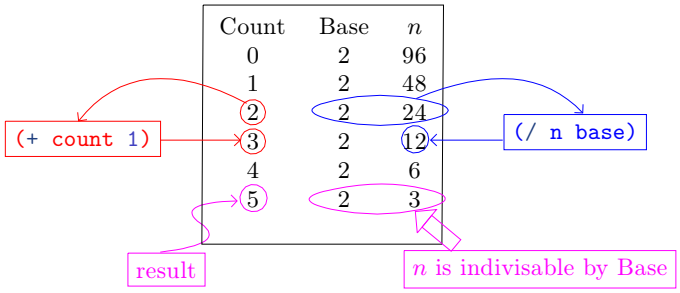


**Table 1.** The Evoluation of Process in Evaluating `(extract-expt 96 2)`

`(extract-expt 96 2)`, we can devise this procedure a little bit more intuitively:

```
(define (extract-expt n base)
  (define (extr-iter count n base)
    (define (indivisable? a b)
      (not (= (remainder b a) 0)))
    (if (indivisable? base n)
        count
        (extr-iter (+ count 1) (/ n base) base)))
  (extr-iter 0 base n))
```

Using `extract-expt` we can readily write the `car` and `cdr` in this representation:

```
(define (car n)
  (extract-expt n 2))

(define (cdr n)
  (extract-expt n 3))
```