Exercise 2.80.

Define a generic predicate `=zero?` that tests if its argument is zero, and install it in the generic arithmetic package. This operation should work for ordinary numbers, rational numbers, and complex numbers.

Answer.

The implementation of generic predicate `=zero?` is much similar to that of `equ?` in exercise 2.79:[1]

```
(define (=zero? x) (apply-generic '=zero? x))
```

Once again, we begin by installing the `=zero?` procedure into the `scheme-number` package:

```
(define (install-scheme-number-package)
  ...
  <other procedures in the package>
  ...
  (put '=zero? '(scheme-number)
       (lambda (x) (= x 0)))
  'done)
```

The package that performs rational arithmetic after installing the `=zero?` predicate appears to be:

```
(define (install-rational-package)
  ;; internal procedures
  ...
  <other internal procedures>
  ...
  (define (=zero-rat? x)
    (= (numer x) 0)

  ;; interface to the rest of the system
  ...
  <other interface procedures>
  ...
  (put '=zero? '(rational)
       (lambda (x) (=zero-rat? x)))
  'done)
```

Following the same way, we can add the `=zero?` predicate into the package that handles complex numbers:[2]

---

1. Another strategy would be by calling the `equ?` procedure with `y` being set to `0`:

```
(define (=zero? x) (equ? x 0))
```

Here we still adopt the practice we've taken in exercise 2.79, simply for the reason that we are required to install the it in the generic arithmetic package.

2. We dispatch the equality test of a complex number with zero from outside world on to the corresponding `=zero?` procedures in terms of rectangular form and polar form still for the sake of additivity. Now add the `=zero?` predicate into both rectangular and polar packages:

```
(define (install-rectangular-package)
  ;; internal procedures
  ...
  (define (=zero? z)
    (and (= (real-part z) 0)
         (= (imag-part z) 0)))

  ;; interface to the rest part of the system
  ...
```

```
(define (install-complex-package)
  ...
  <imported procedures from rectangular and polar packages>
  ...

  ;; internal procedures
  ...
  <other internal procedures>
  ...
  (define (=zero-complex? z)
    (apply-generic '=zero? z))

  ;; interface to the rest of the system
  ...
  <other interface procedures>
  ...
  (put '=zero? '(complex)
       (lambda (z) (=zero-complex? z)))
  'done)
```

```
(put '=zero? '(rectangular)
     (lambda (z) (=zero? z)))
'done)

(define (install-polar-package)
  ;; internal procedures
  ...
  (define (=zero? z)
    (and (= (mangitude z) 0)
         (= (angle z) 0)))

  ;; interface to the rest part of the system
  ...
  (put '=zero? '(polar)
       (lambda (z) (=zero? z)))
  'done)
```