Exercise 2.86.

Suppose we want to handle complex numbers whose real parts, imaginary parts, magnitudes, and angles can be either ordinary numbers, rational numbers, or other numbers we might wish to add to the system. Describe and implement the changes to the system needed to accommodate this. You will have to define operations such as `sine` and `cosine` that are generic over ordinary numbers and rational numbers.

Answer.

To make the system accomodate this mutability, we'd better revisit the complex package we saw in section 2.5.1 and find out what traps it in only competent with those complex numbers whose components are of ordinary numbers. Well, the reason is that all the arithmetic operations between the component of complex numbers (such as `add-complex`) are implemented using plain arithmetic operations on scheme number (such as `+`). This inspire us to update the complex number system by expressing the notion of addition, subtraction, multiplication and division between the components of rational numbers using generic procedures.

```
(define (install-complex-package)
  ;; imported procedures from rectangular and polar packages
  (define (make-from-real-imag x y)
    ((get 'make-from-real-imag 'rectangular) x y))
  (define (make-from-mag-ang r a)
    ((get 'make-from-mag-ang 'polar) r a))

  ;; internal procedures
  (define (add-complex z1 z2)
    (make-from-real-imag (add (real-part z1) (real-part z2))
                         (add (imag-part z1) (imag-part z2))))
  (define (sub-complex z1 z2)
    (make-from-real-imag (sub (real-part z1) (real-part z2))
                         (sub (imag-part z1) (imag-part z2))))
  (define (mul-complex z1 z2)
    (make-from-mag-ang (mul (magnitude z1) (magnitude z2))
                       (add (angle z1) (angle z2))))
  (define (div-complex z1 z2)
    (make-from-mag-ang (div (magnitude z1) (magnitude z2))
                       (sub (angle z1) (angle z2))))

  ;; interface to the rest of the system
  (define (tag x) (attach-tag 'complex x))
  (put 'add '(complex complex)
       (lambda (x y) (tag (add-complex x y))))
  (put 'sub '(complex complex)
       (lambda (x y) (tag (sub-complex x y))))
  (put 'mul '(complex complex)
       (lambda (x y) (tag (mul-complex x y))))
  (put 'div '(complex complex)
       (lambda (x y) (tag (div-complex x y))))
  (put 'make-from-real-imag 'complex
       (lambda (x y) (tag (make-from-real-imag x y))))
  (put 'make-from-mag-ang 'complex
       (lambda (r a) (tag (make-from-mag-ang r a))))
  'done)
```

The operations like `sine` and `cosine` that are generic over ordinary numbers and rational numbers are define as follows:

```
(define (sine x) (apply-generic 'sine x))
```

```
(define (cosine x) (apply-generic 'cosine x))
```

We then add these procedures to the scheme number and rational packages:

```
(define (install-scheme-number-package)
  ...
  <other procedures in the package>
  ...
  (put 'sine 'scheme-number
       (lambda (x) (tag (sin x))))
  (put 'cosine 'scheme-number
       (lambda (x) (tag (cos x))))
  'done)

(define (install-rational-package)
  ;; internal procedures
  ...
  <other internal procedures>
  ...

  ;; interface to the rest of the system
  ...
  <other interface procedures>
  ...
  (put 'sine 'rational
       (lambda (x) (tag (sin x))))
  (put 'sine 'rational
       (lambda (x) (tag (cos x))))
  'done)
```