Exercise 2.78.

The internal procedures in the `scheme-number` package are essentially nothing more than calls to the primitive procedures `+`,`-`, etc. It was not possible to use the primitives of the language directly because our type-tag system requires that each data object have a type attached to it. In fact, however, all Lisp implementations do have a type system, which they use internally. Primitive predicates such as `symbol?` and `number?` determine whether data objects have particular types. Modify the definitions of `type-tag`, `contents`, and `attach-tag` from section 2.4.2 so that our generic system takes advantage of Scheme's internal type system. That is to say, the system should work as before except that ordinary numbers should be represented simply as Scheme numbers rather than as pairs whose `car` is the symbol `scheme-number`.

Answer.

```
(define (attach-tag type-tag contents)
  (if (eq? type-tag 'scheme-number)
      contents
      (cons type-tags contents)))

(define (type-tag datum)
  (cond ((number? datum) 'scheme-number)
        ((pair? datum) (car datum))
        (else
          (error "Bad tagged datum -- TYPT-TAG" datum))))

(define (contents datum)
  (cond ((number? datum) datum)
        ((pair? datum) (cdr datum))
        (else
          (error "Bad tagged datum -- CONTENTS" datum))))
```

*. Creative Commons  2013, Lawrence R. Amlord(颜世敏).