

- 9a:** [HKK 9.5] Rewrite `list->sequence` so that it has a branch for sequence reference.
- 9b:** [Variant on HKK 9.20] Global Amalgamations Corp. has just acquired yet another smaller company and is busily integrating the data processing operations of the acquired company with that of the parent corporation. Luckily, both companies are using Scheme, and both have set up their operations to tolerate multiple representations. Unfortunately, one company uses operation tables as type tags, and the other uses procedural representations (i.e., message passing). Thus, not only are multiple representations now co-existing, but some of them are type-tagged data and others are message-passing procedures. You have been called in as a consultant to untangle this situation.

What is the minimum that needs to be done to make the two kinds of representation happily coexist? Illustrate your suggestion concretely using Scheme as appropriate. You may want to know that there is a built-in predicate `pair?` that tests whether its argument is a pair, and a similar one, `procedure?`, that tests whether its argument is a procedure.

As an example, the database might have been created using:

```
(list
  (tagged-datum 'movie (make-movie ...)) ;; A movie from the company using type tags
  (lambda (op)                               ;; This one, from the other company,
    (cond ((equal? op 'title) ...)          ;; uses message-passing like on page 247
          ...))
  (tagged-datum 'cd (make-cd ...))          ;; A cd using type tags
  ...
)
```

You need to write accessors and predicates (like those on page 256) which work with any item from the database.

- 9c:** [Variant on HKK 9.22] Assume that `infinity` has been defined as a special number that is greater than all normal (finite) numbers and that when added to any finite number or to itself, it yields itself. (In some Scheme systems you can define it as follows: `(define infinity (/ 1.0 0.0))`.) Now there is no reason why sequences need to be of finite length.
- (a) Write a constructor for some interesting kind of sequence which is infinite in length.
 - (b) Demonstrate that your infinite sequence can be used together with `sequence-cons` or Exercise 9.6's `sequence-map` to produce a new infinite sequence.