Exercise 2.47.

Here are two possible constructors for frames:

```scheme
(define (make-frame origin edge1 edge2)
  (list origin edge1 edge2))

(define (make-frame origin edge1 edge2)
  (cons origin (cons edge1 edge2)))
```

For each constructor supply the appropriate selectors to produce an implementation for frames.

Answer.

To give the corresponding selectors of these representations, we're supposed to view them in a intuitive way.

Probably, we may feel comfortable with the first `make-frame` procedure, which represents the frame as a list, as shown in Figure 1. This arouses us to implement the selectors as:

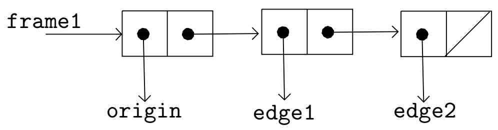

Figure 1. The frame represented as a list

```scheme
(define (origin-frame frame)
  (car frame))

(define (edge1-frame frame)
  (cadr frame))

(define (edge2-frame frame)
  (caddr frame))
```

The second `make-frame` procedure, however, fully exploits all the fields of a pair. And it leads to another way to express a frame, as shown in figure 2. Likewise, we can implement the selectors of such

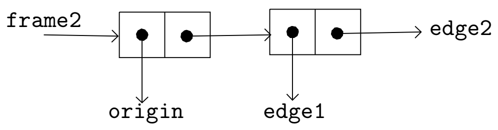

Figure 2. An alternative way to represent a frame

representation in Lisp:

```scheme
(define (origin-frame frame)
  (car frame))

(define (edge1-frame frame)
  (cadr frame))

(define (edge2-frame frame)
  (cddr frame))
```