Exercise 3.33.

Ben Bitdiddle tests the lazy list implementation given above by evaluating the expression

```
(car '(a b c))
```

To his surprise, this produces an error. After some thought, he realizes that the "lists" obtained by reading in quoted expressions are different from the lists manipulated by the new definitions of `cons`, `car`, and `cdr`. Modify the evaluator's treatment of quoted expressions so that quoted lists typed at the driver loop will produce true lazy lists.

Answer.

The original version of `eval`, the text been quoted is directly returned when encounters a quoted expression. For example, evaluating the expression `'(a b c)` obtains `(a b c)`. Hence, by the new definitions of `cons`, `car`, and `cdr`, to response Ben's request, the evaluator applied the resulting list to its arguments, which obviously runs into error.

```
(car '(a b c))
;Unknown procedure type -- APPLY (a b c)
```

So what we desire is the lazy evaluator transforms the problem of evaluating the expression

```
'(a b c)
```

to the problem of evaluating the following expression involving `cons` and `quote` expressions:

```
(cons (quote a)
      (cons (quote b)
            (cons (quote c)
                  (quote ()))))
```

For the latter one can be correctly handled by the new definition of `cons`, `car`, and `cdr`.

To do this, we should enable the evaluator to dispatch correctly on Lisp pair and plain quoted text. The `quoted?` clause of `eval` becomes

```
((quoted? exp) (quoted-value exp env))
```

This is almost the same as the `quoted?` clause of `eval` in section 4.1.1. For lazy evaluation, however, we need to evaluate the equalvalent quoted list in the current environment to construct lazy pairs:

```
(define (quoted-value exp env)
  (let ((quoted-text (cadr exp)))
    (if (pair? quoted-text)
        (eval (quoted-list->cons quoted-text) env)
        quoted-text)))

(define (quoted-list->cons exp)
  (cond ((null? exp) (list 'quote ()))
        ((not (pair? exp)) exp)
        (else
         (list 'cons
               (list 'quote (car exp))
               (quoted-list->cons (cdr exp))))))
```


Email address: informlarry@gmail.com