Exercise 4.14.

Eva Lu Ator and Louis Reasoner are each experimenting with the metacircular evaluator. Eva types in the definition of `map`, and runs some test programs that use it. They work fine. Louis, in contrast, has installed the system version of `map` as a primitive for the metacircular evaluator. When he tries it, things go terribly wrong. Explain why Louis's `map` fails even though Eva's works.

Answer.

Eva's `map` works because it is written in the language been implemented and our evaluator program reduces expressions ultimately to the application of primitive procedures whose reliability is guaranteed.

Louis's plan failed because he mixed up the language been implemented and the implementation language. The metacircular representation of procedures might not be the same as that of the underlying Scheme. For example, the compound procedure `abs` in section 1.1.6 is represented as

```
(list 'procedure
      '(x)
      '(cond ((< x 0) (- x))
             ((= x 0) 0)
             (else x))
      the-global-environment)
```

in the metacircular evaluator, whereas the underlying Scheme uses an alternative notation:

```
;Value 29: #[compound-procedure 29 abs]
```

So when the evaluator striped off the expression `(map abs (list -10 2.5 -11.6 17))` and finally reached:

```
(apply map                                    ;; the underlying Scheme APPLY
       (list (list 'procedure '(x) '(cond ...) the-global-environment)
             (list -10 2.5 -11.6 17)))
```

We see that the compound procedure `abs` is not organized in the form for which `map` expected. The evaluator is therefore crashed for this piece of inconsistence.

*. Creative Commons 2014, Lawrence X. Amlord (颜世敏, aka 颜序).
Email address: informlarry@gmail.com