CS 61A, Summer 2006 August 2, 2006

HOMEWORK ASSIGNMENT 6B

DUE WEDNESDAY AUGUST 9, 2006 AT 11:59 PM

A version of the metacircular evaluator is online in `~cs61a/lib/mceval.scm`
A version of the analyzing evaluator is online in `~cs61a/lib/analyze.scm`

1. Exercises 4.3, 4.6, 4.7, 4.10, 4.11, 4.13, 4.14, 4.15, 4.22, 4.23, 4.24

2. Modify the metacircular evaluator to allow *type-checking* of arguments to procedures. Here is how the feature should work. When a new procedure is defined, a formal parameter can be either a symbol as usual or else a list of two elements. In this case, the second element is a symbol, the name of the formal parameter. The first element is an expression whose value is a predicate function that the argument must satisfy. That function should return `#t` if the argument is valid. For example, here is a procedure `foo` that has type-checked parameters `num` and `list`:

```
> (define (foo (integer?  num) ((lambda (x) (not (null?  x))) list))
(nth num list))
FOO
> (foo 3 '(a b c d e))
D
> (foo 3.5 '(a b c d e))
Error:  wrong argument type -- 3.5
> (foo 2 '())
Error:  wrong argument type -- ()
```

In this example we define a procedure `foo` with two formal parameters, named `num` and `list`. When `foo` is invoked, the evaluator will check to see that the first actual argument is an integer and that the second actual argument is not empty. The expression whose value is the desired predicate function should be evaluated with respect to `foo`'s defining environment. (Hint: Think about `extend-environment`.)