Exercise 4.24.

Design and carry out some experiments to compare the speed of the original metacircular evaluator with the version in this section. Use your results to estimate the fraction of time that is spent in analysis versus execution for various procedures.

Answer.

We extend the `driver-loop` in section 4.1.4 to show the time the metacircular evaluator takes to evaluate a expression.

```
(define time-prompt ";;; Elapsed time:")

(define (driver-loop)
  (prompt-for-input input-prompt)
  (let ((input (read)))
    (define start-time (runtime))
    (let ((output (eval input the-global-environment)))
      (announce-output output-prompt)
      (user-print output)
      (announce-output time-prompt)
      (user-print (- (runtime) start-time))))
  (driver-loop))
```

For conciseness, we perform benchmark on the familiar program to compute Fibonacci numbers:

```
(define (fib n)
  (cond ((= n 0) 1)
        ((= n 1) 1)
        (else (+ (fib (- n 1))
                 (fib (- n 2))))))
```

Now we start the driver loop and see the interaction with the original metacircular evaluator, :

```
;;; M-Eval input:
(fib 10)

;;; M-Eval value:
89
;;; Elapsed time:
.04000000000000001

;;; M-Eval input:
(fib 16)

;;; M-Eval value:
1597
;;; Elapsed time:
.65

;;; M-Eval input:
(fib 20)

;;; M-Eval value:
10946
;;; Elapsed time:
4.23
```

```
;;; M-Eval input:
(fib 24)

;;; M-Eval value:
75025
;;; Elapsed time:
28.529999999999998
```

And here is the interaction with the metacircular evaluator which seperate syntactic analysis from execution:

```
;;; M_Eval_2e input:
(fib 10)

;;; M_Eval_2e value:
89
;;; Elapsed time:
.01999999999999602

;;; M_Eval_2e input:
(fib 16)

;;; M_Eval_2e value:
1597
;;; Elapsed time:
.37000000000000455

;;; M_Eval_2e input:
(fib 20)

;;; M_Eval_2e value:
10946
;;; Elapsed time:
2.460000000000001

;;; M_Eval_2e input:
(fib 24)

;;; M_Eval_2e value:
75025
;;; Elapsed time:
16.79
```

Since the optimized evaluator hardly spent any effort to analyze the syntax of an expression, we can approximately regard the time it runs as the execution time of an expression. Consider the results above, the original metacircular evaluator approximately spents

$$\left(1-\frac{0.20+0.37+2.46+16.79}{0.04+0.65+4.23+28.53}\right)\times 100\% = 59\%$$

of its efforts to analyze the syntax of expressions. Execution only takes up 41% of its work. This explains why our original version of metacircular evaluator is inefficient.