

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Department of Electrical Engineering and Computer Science

6.001, Fall 2007

Structure and Interpretation of Computer Programs

The 6.001 Final Exam will be held on December 19, 2007, in 32-141 at 9:00AM--12:00Noon. The exam is open book. You may bring any books or papers that you might want to the quiz, but you may not use a computer or communications device during the quiz.

Notice:

Recently, the workers planning to demolish 10-250 in the spring, to make a new "better" lecture hall, discovered a cache of strange documents buried in the floor under the room. There were some fragments of code that appeared to perhaps be of some interest to us. While the 6.001 staff denies any knowledge of these materials, they may have some bearing on the contents of the upcoming exam. We suggest that you examine this material very closely in the days before the exam. And be sure to BRING IT WITH YOU to the exam!

2/5

Date: December 03, 3007
From: Quirky Thomas, Architect
To: Rosiebed Peters, Assistant Registrar
Re: 10-250 An Important Archeological Find?

As part of my assignment to design a new ampitheatre to replace the venerable 10-250 I was nosing around the old site. Apparently I wandered through the "infinite corridor" until I reached the weird ruins of the Stata Center. (That structure seems to have collapsed before it was ever inhabited!) Among the usual detritus found in habitations of Homo eecs-academicus, such as broken coffee cups, fragments of chalk, pieces of pencils, floppy disks, and RAM chips, I observed some packets of ancient paper documents. I was careful not to disturb the venue and I called in our resident Academic Archeologist, Professor Steinhur Artberg to examine the material.

Professor Artberg has done a preliminary site survey. He has determined that some of the fragments can be accurately dated to work done at this establishment almost exactly 1000 years ago. The paper is quite delicate and had to be chemically treated to make it possible for him to unfold any of it without it turning immediately into dust.

Some of the material seems to be fragments of code written in an extinct language called S&=eme, designed for primitive computing engines that work using electromagnetic interactions, without employing positronic fields! It is not apparent how those ancient engineers managed to get anything done with such primitive mechanisms, but analysis of these materials may give us some insight into the amazing accomplishments of our ancestors.

At 9:00AM on December 19, Professor Artberg will meet with Professor Bitdiddle's staff of crack programmers to try to get some insight into this material. I enclose copies of some of the fragments we found for your perusal.

THIS BADLY-DAMAGED FRAGMENT WAS FOUND BEHIND THE CHALKBOARD PANEL

```
(define (eval exp env)
  (cond ((self-evaluating? exp) exp)
        ((variable? exp) (lookup-variable-value exp env))
```

MISSING STUFF HERE -- COVERED WITH BITUMINOUS MATERIAL

```
((lambda? exp)
  (make-lambda-procedure (lambda-parameters exp)
                          (lambda-body exp)
                          env))

((alpha? exp)
  (make-alpha-procedure (alpha-parameters exp)
                        (alpha-body exp)))
```

MISSING STUFF HERE -- APPARENTLY CHEWED BY A RODENT

```
((application? exp)
  (apply (eval (operator exp) env)
         (list-of-values (operands exp) env)
         env))

(else
  (error "Unknown expression type -- EVAL" exp))))
```

```
(define (apply procedure arguments env)
  (cond ((primitive-procedure? procedure)
        (apply-primitive-procedure procedure
                                     arguments))
        ((lambda-procedure? procedure)
        (eval-sequence
         (procedure-body procedure)
         (extend-environment
          (procedure-parameters procedure)
          arguments
          (procedure-environment procedure))))
        ((alpha-procedure? procedure)
        (eval-sequence
         (procedure-body procedure)
         (extend-environment
          (procedure-parameters procedure)
          arguments
          env)))
        (else
         (error "Unknown procedure type -- APPLY"
                procedure))))
```

THIS FRAGMENT WAS ALMOST PERFECTLY PRESERVED!

```

((env)
 (val)
 ((assign val (op make-compiled-procedure) (label entry20) (reg env))
  (goto (label after-lambda19))
  entry20
  (assign env (op compiled-procedure-env) (reg proc))
  (assign env
   (op extend-environment) (const (f g)) (reg arg1) (reg env))
  (assign val (op make-compiled-procedure) (label entry22) (reg env))
  (goto (reg continue))
  entry22
  (assign env (op compiled-procedure-env) (reg proc))
  (assign env (op extend-environment) (const (x)) (reg arg1) (reg env))
  (assign proc (op lookup-variable-value) (const f) (reg env))
  (save continue)
  (save proc)
  (assign proc (op lookup-variable-value) (const g) (reg env))
  (assign val (op lookup-variable-value) (const x) (reg env))
  (assign arg1 (op list) (reg val))
  (test (op primitive-procedure?) (reg proc))
  (branch (label primitive-branch25))
  compiled-branch24
  (assign continue (label after-call23))
  (assign val (op compiled-procedure-entry) (reg proc))
  (goto (reg val))
  primitive-branch25
  (assign val (op apply-primitive-procedure) (reg proc) (reg arg1))
  after-call23
  (assign arg1 (op list) (reg val))
  (restore proc)
  (restore continue)
  (test (op primitive-procedure?) (reg proc))
  (branch (label primitive-branch28))
  compiled-branch27
  (assign val (op compiled-procedure-entry) (reg proc))
  (goto (reg val))
  primitive-branch28
  (assign val (op apply-primitive-procedure) (reg proc) (reg arg1))
  (goto (reg continue))
  after-call26
  after-lambda21
  after-lambda19
  (perform (op define-variable!) (const compose) (reg val) (reg env))
  (assign val (const ok))))

```

5/5

THIS TINY, BROKEN FRAGMENT MAY BE A CRITICAL CLUE!

```
(alpha (self)
  (let ((x 0) (y 0))
    (lambda (message)
      (cond ((eq? message 'lkjfr rhjewa  INSECT DAMAGE!
        (set! y (froblicate x y)))
```

THE REST OF THIS WAS GONE, BUT THERE WAS A LITTLE BIT MORE:

```
(define (froblicate x-coord y-coord)
  ((ask self 'update-y) x-coord y-coord))
```

Anyway, there are numerous other fragments that we can recover.
I will bring them to the meeting on 19 December.

Steinhur Artberg