

MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
Department of Electrical Engineering and Computer Science  
6.001—Structure and Interpretation of Computer Programs  
Fall Semester, 2006

**Final Exam – Solutions**

**Part 1 (32 points): Evaluators**

**Question 1:**

Abstractions are:

`(cadr exp)`

`(caddr exp)`

**3 points for each answer – total of 6**

**Question 2:**

```
(define (eval-let* exp env)
  (let*-aux (let*-clauses exp) (let*-body exp) env))

(define (let*-aux clauses body env)
  (if (null? clauses)
      (eval-sequence body env)
      (let ((new-env
             (extend-environment (car (car clauses))
                                (m-eval (cadr (car clauses)) env)
                                env))))
        (let*-aux (cdr clauses) body new-env))))
```

**Total of 13 points**

- Eval-sequence – 2 points, 1 if just eval
- evaluating body – 1 point
- new environment – 2 points
- getting variable – 2 points
- getting value, including eval – 4 points
- recursive call – 2 points

**Question 3**

```
(define (let*->let exp)
  (let*-help (let*-clauses exp) (let*-body exp)))

(define (let*-help clauses body)
  (if (null? (cdr clauses))
      (cons 'let (cons clauses body))
      (append (list 'let (list (car clauses)))
              (list (let*-help (cdr clauses) body))))))
```

**Total of 14 points**

- **checking for base case – 2 points**
- **creating last let – 2 points for parts, 2 points for right structure**
- **recursive step**
  - **recursive call – 1 point**
  - **wrapping list around – 1 point**
  - **creating let and clause – 3 points**
  - **gluing parts together – 2 points**

**Part 2 (20 points): Models of Evaluation**

**Question 4:**

In ordinary evaluator

**(14 9 5 2)**

In lazy, non-memoized evaluator

**(5 4 3 2)**

**5 points for each answer – 2 points if order reversed**

**Question 5:**

Suppose that `proc1` and `proc2` are being applied asynchronously, with no serialization. What is the set of possible values for `x` at the completion of both evaluations?

**5, 9, 11, 15, 25**

**Question 6:** Suppose that the `set!` expressions in both `proc1` and `proc2` in the previous question are serialized by the same serializer, and that the two procedures are being applied asynchronously. What is the set of possible values for `x` at the completion of both evaluations?

**11, 25**

**5 points for each question**

**Part 3 (28 points): Data structures****Question 7:**

```
(define (add-poly p1 p2)
  (cond ((null? p1) p2)
        ((null? p2) p1)
        (else (cons (+ (car p1) (car p2))
                      (add-poly (cdr p1) (cdr p2))))))
```

**7 points total – 2 for base cases, 5 for recursive call****Question 8:**

```
(define (eval-poly p x)
  (poly-aux p x (length p)))

(define (poly-aux p x n)
  (foldr ANSWER5 ANSWER6
         (map1 ANSWER7 p)))
```

Solution is:

```
(define (poly-aux p x n)
  (foldr + 0
         (map1 (lambda (l) (* (car l) (expt x (- n (length l)))))
               p)))
```

**9 points total – 2 for +; 1 for 0; 6 for lambda expression****Question 9:**Using these ideas (including `add-poly`), implement polynomial multiplication:

```
(define (mult-poly p1 p2)
  (foldr ANSWER8 ANSWER9
         (map1 ANSWER10 p1)))
```

Solution is:

```
(define (mult-poly p1 p2)
  (foldr add-poly '()
         (map1 (lambda (x) (append (pad-poly (- (length p1) (length x)))
                                   (scale-poly p2 (car x))))
               p1)))
```

**12 points total – 3 for `add-poly`; 2 for `'()`; 7 for lambda expression**

**Part 4 (15 points): Environment diagrams**

**1 point for each answer**

**Question 10:** For each of the following symbols, bound in the indicated environment, indicate the value to which it is bound.

**p3, p2, p1, 7, 5**

**Question 11:** For each of the following procedure objects, indicate the enclosing environment:

**GE, E5, GE, E6**

**Question 12:** Indicate the enclosing environment for each environment:

**GE, GE, E5, E5, E6, GE**

**Part 5 (16 points): General questions**

For each of the following, indicate if the statement is **true** or **false**. Grading on this part will be 2 points for each correct answer, -1 point for each incorrect answer, and 0 points for no answer.

**Answers are: F, F, F, T, T, T, F, F**

**Part 6 (15 points): Basic procedures**

**Question 21:** Complete the definition of the procedure `foldr` using only basic Scheme procedures.

```
(define (foldr op init lst)
  (if (null? lst)
      init
      (op (car lst) (foldr op init (cdr lst)))))
```

**6 points** – 1 for end test, 1 for init, 4 for recursive call

**Question 22:** Write a procedure `map-rev`, which behaves like `map` except that the returned value has the elements in the reverse order to the input list. Do this without using the procedure `reverse`, and using only basic Scheme.

```
(define (map-rev op lst)
  (define (help done todo)
    (if (null? todo)
        done
        (help (cons (op (car todo)) done)
              (cdr todo))))
  (help '() lst))
```

**9 points** – 1 for end test, 2 for return, 6 for iterative call

**Part 7 (18 points): Orders of Growth**

**3 points for each part**

**Question 23:**

quadratic linear quadratic

**Question 24:**

linear, constant, linear

**Question 25:**

exponential, log, exponential

**Part 8 (25 points): Mutation**

**Question 26:**

```
(define (shift-right ring)
  (set-car! (cdr ring) (cdr (cadr ring))))
```

**7 points – 3 for new element, 2 for (cdr ring), 2 for set-car!**

```
(define (shift-left ring)
  (let ((start (cadr ring)))
    (set-car! (cdr ring) (find-previous start start))))
```

**8 points – 4 for find-previous call, 2 for (cdr ring), 2 for set-car!**

**Question 27:**

```
(define (insert-into-ring element ring)
  (let ((new-cell (list element)))
    (set-cdr! new-cell (cdr (cadr ring)))
    (set-cdr! (cadr ring) new-cell)))
```

**10 points – 3 for new cell; 3 for each mutation**

**Part 9. (31 points): Object oriented systems**

**Question 28:**

prints I'm going to watch the Red Sox! in an infinite loop – 4 points

**Question 29:**

```
I've got too many psets!  
eric says I have writer's block!  
I'm going to watch the Red Sox!
```

9 points, 2 for first line, 5 for second line (name has to be correct); 2 for last line

**Question 30:**

```
I've got too many psets!  
I'm going to watch the Red Sox!
```

5 points

**Question 31:**

```
I'm going to watch the Red Sox
```

4 points

**Question 32:**

```
eric
```

3 points

**Question 33:**

```
does anyone really have a name  
all-names-are-ephemeral
```

6 points