

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Spring Semester, 2007

Quiz I – Solutions

Part 1: (20 points)

Question 1:

```
((lambda (a b)
  (- (/ a b)
    (+ a b)))
  8 4)
```

Value: -10 Type: number

Question 2:

```
((lambda (a + b)
  (+ a b))
  3 * 4)
```

Value: 12 Type: number

Question 3:

```
((lambda (x)
  (lambda (y)
    (expt x y)))
  3)
```

Value: compound procedure Type: number -> number

Question 4:

```
(lambda (a b) (* 2 a))
```

Value: compound procedure Type: number, A -> number

Question 5:

```
(define x 3)
(define y 4)
((lambda (x) (* x y)) 2)
```

Value: 8 number

Part 2: (22 points)

First, the elements of a bracket are a list of entries, created using:

```
(define (make-entry team seed ranking)
  (list team seed ranking))
```

Question 6: Write an implementation for the accessors to complete this data abstraction

```
(define entry-team car) ;; or first

(define entry-seed cadr) ;; or second

(define entry-ranking caddr) ;; or third
```

Question 7:

```
(define (run-a-round bracket probable-winner)
  ;; okay to assume that there are an even number of elements in bracket
  (if (null? bracket)
      bracket
      (cons (probable-winner (car bracket) (cadr bracket))
            (run-a-round (cddr bracket) probable-winner))))
```

Rewrite `run-a-round` in iterative form. Be sure that the result preserves the order that you saw with the recursive version (you may want to use `reverse`, which is a builtin Scheme procedure that reverses the elements of a list).

```
(define (run-a-round bracket probable-winner)
  (define (r-iter done todo probable-winner)
    (if (null? todo)
        (reverse done)
        (r-iter (cons (probable-winner (car todo) (cadr todo))
                      done)
                (cddr todo)
                probable-winner)))
  (r-iter '() bracket probable-winner))
```

Question 8:

```
(define (run-all-rounds bracket probable-winner)
  (if (null? (cdr bracket))
      (car bracket)
      (run-all-rounds (run-a-round bracket probable-winner) probable-winner)))
```

Part 3: (18 points)

Question 9:

```
(define winner-ranking (lambda (team1 team2)
  (if (>= (random)
        (/ (entry-ranking team2)
            (+ (entry-ranking team1) (entry-ranking team2))))
      team2
      team1)))
```

Question 10:

```
(define (generate-winner accessor)
  (lambda (team1 team2)
    (if (>= (random)
          (/ (accessor team2)
              (+ (accessor team1) (accessor team2))))
        team2
        team1)))
```

Part 4: (22 points)

Question 11: Using some combination of `map` and/or `filter`, write an implementation of `get-elts`.

```
(define (get-elts selector data)
  (map selector data))
```

Question 12: We want you to complete the code for `build-hist`, which will complete the data abstraction. Note that the parameter `same?` in `build-hist` will be used to compare keys (in the example above we used `string=?` which compares two strings to decide if they are the same), and `elts` is a list of keys.

```
(define (build-hist elts same?)
  (if (null? elts)
      empty-histogram
      (let ((next (car elts)))
        (adjoin-histogram
         (histogram-element next
          (LENGTH (FILTER (LAMBDA (X) (SAME? NEXT X)) ELTS)))
         (build-hist
          (FILTER (LAMBDA (X) (NOT (SAME? X NEXT))) (CDR ELTS))
          same?))))))
```

Part 5: (18 points)

For the questions below, choose your answer from:

- A: constant
- B: linear
- C: exponential
- D: quadratic
- E: logarithmic
- F: something else

Question 13: What is the order of growth in time of the procedure `get-max` measured in terms of the length of the histogram?

B linear

What is the order of growth in space of the procedure `get-max`, measured as the maximum number of deferred operations, as a function of the length of the histogram?

A constant

Question 14: What is the order of growth in time of the procedure `remove` measured in terms of the length of the histogram?

B linear

What is the order of growth in space of the procedure `remove`, measured as the maximum number of deferred operations, as a function of the length of the histogram?

B linear

Question 15: What is the order of growth in time of the procedure `sort-hist` measured in terms of the length of the histogram?

D quadratic

What is the order of growth in space of the procedure `sort-hist`, measured as the maximum number of deferred operations, as a function of the length of the histogram?

B linear