MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.090—Building Programming Experience
IAP, 2007

Quiz I Solutions

Open Book – no computing devices

Separately, we have distributed an answer sheet. You may use the space on the exam booklet for whatever temporary work you find useful, but you **MUST** enter your answers into the answer sheet. **Only the answer sheet will be graded.** Each problem that requires an answer has been numbered. Place your answer at the corresponding number in the answer sheet.

Note that any procedures or code fragments that you write will be judged not only on correct function, but also on clarity and good programming practice.

The answer sheet asks for your section number and tutor. For your reference, here is the table of section numbers.

Section	Time	Room	Instructor	Tutor
R01	MTWRF 1 PM	32-141	Jacob Strauss	Jacob Strauss

Part 1: (24 points)

For each of the following expressions or sequences of expressions, state the value returned as the result of evaluating the final expression in each set, or indicate that the evaluation results in an error. If the expression does not result in an error, also state the "type" of the returned value, using the notation from lecture. If the result is an error, state in general terms what kind of error (e.g. you might "error: wrong type of argument to procedure"). If the evaluation returns a built-in procedure, write `primitive procedure`, and it's type. If the evaluation returns a user-created procedure, write `compound procedure`, and it's type.

You may assume that evaluation of each sequence takes place in a newly initialized Scheme system.

For example, for the expression `88` your answer would be

Value:	88
Type:	number

```
(3 * 5)
```

Question 1: Value & Type

error: 3 is not a procedure

```
((lambda (a b) (a b))
 (lambda (c) (* 2 c))
 10)
```

Question 2: Value & Type

20, number

```
(lambda (m n)
  (if m (* 2 n) (/ 2 n)))
```

Question 3: Value & Type

compound procedure, $(boolean, number) \rightarrow number$

```
(let ((x <)
      (y 10)
      (z 20))
  (z y x))
```

Question 4: Value & Type

error, 20 is not a procedure

```
((lambda (arg) (arg)) (lambda () 5))
```

Question 5: Value & Type

5, number

```
(define y (list 42 17 54))
(define z (cons 7 (cdr y)))
z
```

Question 6: Value & Type

(7 17 54), list<number>

Part 2: (20 points)

We will consider the following function: $\text{tower}(x, n) = x^{(x^{(x^{\cdots})})}$ (where x appears n times). For example, $\text{tower}(2, 3) = 2^{(2^{(2^1)})} = 16$. By definition $\text{tower}(x, 0) = 1$. You should use `(expt x n)` which computes x^n to help implement `tower` below.

```
(define (tower x n)
  INSERT1)
```

Question 7: What expression should be used for INSERT1?

```
(if (= n 0)
    1
    (expt x
          (tower x (- n 1))))
```

Part 3: (30 points)

Ben Bitdiddle (a common character in 6.001) is implementing a rating tracking system for people playing the card game Spades. This tracking system maintains a count of wins and losses for each player. His first job is to implement a player abstraction that wraps up the player's name, number of wins, and number of losses. He's managed to implement the selectors, but the constructor continues to elude him. Help Ben Bitdiddle out by completing the constructor in such a way that the contract is preserved (ie the selectors return the appropriate values).

```
(define (make-player name wins losses)
  INSERT2)

(define (player-name player)
  (car player))

(define (player-wins player)
  (car (cdr player)))

(define (player-losses player)
  (car (cdr (cdr player))))


; example usage:
(define p (make-player "Ben" 5 3))

(player-name p)
;Value: "Ben"
(player-wins p)
;Value: 5
(player-losses p)
;Value: 3
```

Question 8: What expression should be used for INSERT2?

```
(list name wins losses)
```

————ABSTRACTION BARRIER————

The number of primary interest to the players (and Ben) is their win ratio, which is the ratio of wins to total games.

```
(define (player-win-ratio player)
  INSERT3)
```

Question 9: What expression should be used for INSERT3?

```
(/ (player-wins player)
   (+ (player-wins player)
      (player-losses player)))
```

Finally, given a list of players who have only played each other, for every win counted by one player there should be a loss counted by another. Thus, the total number of wins must equal the total number of losses. We can use this fact to check to see if anyone is cheating and misreporting their won/loss record.

Ben is writing a procedure called `check-records` which takes a list of players and returns 0 if the number of wins equals the number of losses, returns a positive number if there are more wins than losses, and a negative number if there are more losses than wins. Help him finish the procedure.

```
(define (check-record list-of-players)
  (if (null? list-of-players)
      INSERT4
      (+ INSERT5
         (check-record (cdr list-of-players)))))
```

Question 10: What expression should be used for INSERT4?

0

Question 11: What expression should be used for INSERT5?

```
(- (player-wins
    (car list-of-players))
   (player-losses
    (car list-of-players)))
```

Part 4: (12 points)

Indicate whether the following procedures are iterative or recursive.

```
(define (slow-add x y)
  (if (= x 0)
      y
     (slow-add (- x 1) (+ y 1))))

(define (slow-mul x y)
  (if (= x 0)
      0
      (+ y (slow-mul (- x 1) y))))

(define (watch-this n)
  (if (= n 0)
      1
      (if (< n 0)
          (watch-this (- n))
          (* n (watch-this (- n 1))))))

(define (excitement)
  (excitement))
```

Question 12: Is `slow-add` iterative or recursive?

Iterative

Question 13: Is `slow-mul` iterative or recursive?

Recursive

Question 14: Is `watch-this` iterative or recursive?

Recursive

Question 15: Is `excitement` iterative or recursive?

Iterative

Part 5: (14 points)

Consider the following procedures:

```
(define (comp f g)
  (lambda (x) (f (g x))))

(define (repeat f n)
  (if (= n 1)
      f
      TO-BE-COMPLETED))
```

The goal of `repeat` is to return a procedure of one argument that has the effect, when applied to an argument, of recursively applying the argument `f` to the previous result `n` times. Here are two possible completions to the missing portion of `repeat`:

A. `(comp f (repeat f (- n 1)))`

B. `(comp (repeat f (- n 1)) f)`

Question 16: Which of the following best describes the behavior of these choices

1. Only option A will work as described
2. Only option B will work as described
3. Both option A and B will work as described
4. Neither option A nor B will work as described

Both option A and B will work as described

Assuming that `repeat` is correctly coded, use it to complete the following procedure so that it performs multiplication by successive addition. In other words, `(mul a b)` should return the product of `a` and `b` using only the primitive operation of addition.

```
(define (mul a b)
  INSERT6)
```

Question 17: What expression should be used for INSERT6?

```
((repeat (lambda (x) (+ a x)) b) 0)
```

Assuming that `repeat` is correctly coded, use it to complete the following procedure so that it performs exponentiation by successive multiplication. In other words, `(my-exp a b)` should return the a^b using only the primitive operation of multiplication.

```
(define (my-exp a b)
  INSERT7)
```

Question 18: What expression should be used for INSERT7?

```
((repeat (lambda (x) (* a x)) b) 1)
```