MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.090—Building Programming Experience
IAP, 2005

Quiz I

Closed Book – one sheet of notes

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		Instructor	Tutor
R01	MTWRF 10 AM		Ben Vandiver	Ben Vandiver

Part 1: (18 points)

For each of the following expressions you are to (1) draw the box and pointer diagram corresponding to the list or pair structure created, and (2) write the printed result of evaluating the last expression in the sequence.

```
(define a (list 1 (cons 2 nil) 3))
a
```

Question 1: Draw box and pointer for `a`

Question 2: Printed result

```
(define x (list 1 2))
(define y (list 3 (list 4)))
(define z (append x y))
z
```

Question 3: Draw box and pointer for `z`

Question 4: Printed result

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

Question 5: Draw box and pointer for `z`

Question 6: Printed result

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?

Part 3: (30 points)

Ben Bitdiddle (a common character in 6.001 with no connection to your instructor) 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)
  INSERT1)

(define (player-name player)
  (first player))

(define (player-wins player)
  (second player))

(define (player-losses player)
  (third 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 INSERT1?

————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)
  INSERT2)
```

Question 9: What expression should be used for INSERT2?

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)
      INSERT3
      (+ INSERT4
         (check-record (cdr list-of-players)))))
```

Question 10: What expression should be used for INSERT3?

Question 11: What expression should be used for INSERT4?

Part 4: (12 points)

Indicate whether the following procedures are iterative or recursive.

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

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

(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-mul` iterative or recursive?

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

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

Question 15: Is `excitement` iterative or recursive?

Part 5: (20 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 write "error: wrong type of argument to procedure"). If the evaluation returns a built-in procedure, write `primitive procedure`, and its `type`. If the evaluation returns a user-created procedure, write `compound procedure`, and also indicate its `type` using the notation we introduced in class. 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

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

Question 16: Value

Question 17: Type

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

Question 18: Value

Question 19: Type

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

Question 20: Value

Question 21: Type

```
(define (double x) (* 2 x))
(define (check y)
  (if (< (double y) 10)
      "yip"
      (if (> (double y) 6)
          "yay"
          "yuck")))
check
```

Question 22: Value

Question 23: Type

```
(define three
  (lambda (a b c) (* a b c)))
(three (three 1) (three 2) (three 3))
```

Question 24: Value

Question 25: Type