Quiz 6b

1. (3 points) Identify the programming style of the following code snippets.

(define (make-complex real imag)

(define (dispatch op)

(cond ((eq? op 'magnitude)

(sqrt (+ (square real) (square imag))))

((eq? op 'angle) (atan imag real))

(else (error “Unknown op”))))

dispatch)

(define x (make-complex 3 4))

(x ‘magnitude)

Which style is it?

1. Conventional
2. Data-directed
3. Message passing

(define (make-complex real imag)

(cons real imag))

(define (real x) (car x))

(define (imag x) (cdr x))

(define (angle x)

(atan (imag x) (real x)))

(define (magnitude x)

(sqrt (+ (square (real x)) (square (imag x)))))

(define x (make-complex 3 4))

(magnitude x)

Which style is it?

1. Conventional
2. Data-directed
3. Message passing

(define (make-complex real imag)

(attach-tag ‘complex (cons real imag)))

(put ‘real-part ‘(complex) (lambda (z) (car z)))

(put ‘imag-part ‘(complex) (lambda (z) (cdr z)))

(put ‘magnitude ‘(complex) (lambda (z)

(sqrt (+ (square (real-part z))

(square (imag-part z))))))

(define (real-part z) (apply-generic 'real-part z))

(define (imag-part z) (apply-generic 'imag-part z))

(define (magnitude z) (apply-generic 'magnitude z))

(define x (make-complex 3 4))

(magnitude x)

Which style is it?

1. Conventional
2. Data-directed
3. Message passing
4. (4 points)

((lambda (x y) (+ 5 x y))

10

((lambda (x) (\* 2 x)) 1))

(1 points) a) What does the above code return? \_\_\_\_\_\_\_\_

(3 points) b) How does scheme-1 evaluate the expression?

The first call to eval-1 is given. Show the next 3 calls to eval-1.

(eval-1 ‘((lambda (x y) (+ 5 x y))

10

((lambda (x) (\* 2 x)) 1) )

1. (3 points)Let’s build a **message-passing** model of a color!

make-color should take 3 arguments: red, green, blue

A color should respond to the following messages:

‘red : return the red value

‘green: return the green value

‘blue: return the blue value

‘intensity: return the average of red, green, and blue

Here is a sample interaction:

> (define c (make-color 12 1 2))

c

> (c ‘red)

12

> (c ‘green)

1

> (c ‘blue)

2

> (c ‘intensity)

5

Fill in the code to make this happen:

(define (make-color r g b)