Quiz 9a

1. (3 points) Say what Scheme prints, and draw a box-and-pointer diagram. In the case of an error, say “error” and don’t draw the box-and-pointer diagram. If there is a cycle, which would make it print infinitely, write “cycle” (and still draw the box-and-pointer diagram).

(define x (cons (list 1) 2))

(set-cdr! (car x) (cons 3 (cadar x)))

(set-cdr! x 4)

x

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(define y (list 1 2 3))

(set-cdr! (cddr y) y)

(set-cdr! (cdr y) 4)

y

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(define z (cons (cons (cons 1 2) 3) 4))

(set-cdr! (car z) (cons 5 (caar z)))

(set-cdr! (caar z) (cons 6 ‘()))

z

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (4 points) Somehow, all of your data structures have gotten corrupted! All the pairs have been swapped – what should have been in the car is in the cdr instead, and vice versa. Write a procedure fix-data! that takes in some arbitrary structure created by cons, and swaps the car and the cdr for all of the pairs in that structure, **without** creating any new pairs. It must only modify existing pairs. (Hint: Since we are treating the car and cdr similarly here, it is convenient to think of the structure as a tree, so that you make two recursive calls, as in lesson 5.) Space is provided on the back.

> (define x (cons (cons (cons ‘() 5) 4) (cons ‘() 3)))

> (fix-data! x)

> x

((3) 4 5)

1. (3 points) Write the procedure split-vector, which takes as input a vector vec and an index i. It returns two new vectors – one with the elements from index 0 to i-1, and one with the elements from index i+1 onwards. (Notice that the element at index i does not appear in the result.) At this point you should be saying “I can’t return more than one thing!” You’re absolutely right. So, you should instead return a pair of vectors, i.e. something like (cons left-vec right-vec). Helper procedures are okay.

Warning: This question is not going to be nearly as elegant as other procedures you have written. Expect to write more code than usual.

> (define pair (split-vector (vector ‘h ‘e ‘l ‘l ‘o ‘- ‘w ‘o ‘r ‘l ‘d) 5))

> pair

(#(h e l l o) . #(w o r l d))