Quiz 2a

1. (2 + 1 points) What will Scheme print? (If it causes an error, write Error.) Explaining your answer may earn partial credit if your answer is wrong.
   1. ((lambda (foo bar baz) (bar baz foo))

‘(what a wonderful world)

(lambda (x y) (x bf y))

(lambda (x y) (first y)))

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

* 1. (let ((a 5))

(let ((b (\* a 2)))

(+ a b)))

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

1. (3 points) Write the higher order function while, which takes three arguments – pred, fn, and x. The function fn will be applied to x as long as the predicate returns true. The predicate accepts as its argument the current version of x (i.e. x, or whatever it becomes after fn is applied to it multiple times). When the predicate returns false, the current version of x is returned. **By now you should know that we will cut points for bad style!**

> (while (lambda (x) (> (count x) 2)) bf ‘(lets cut this sentence short))

(sentence short)

1. (4 points) We now want to test while in order to make sure it is correct. Write another version of the lower-power-of-two procedure given below using while. **Your solution must consist of a single call to while with arguments of your choice. You may not call any other procedures directly.**  Explaining your solution can help you get partial credit if your answer is wrong.

> (lower-power-of-two 490)

256

> (lower-power-of-two 3)

2

(define (lower-power-of-two n)

(define (helper so-far)

(if (> so-far n)

(/ so-far 2)

(helper (\* 2 so-far))))

(helper 1))

(define (lower-power-of-two-while n)

(while

)

)