6.101 Recitation 22: Week 12 Lisp part 1 Wrap-Up **5/6/24**

This sheet is yours to keep!

Question 2: Write the equivalent Scheme code for the Python program below:

```
# Python code
def foo(x):
    return 2*x**4

def deriv(f, dx):
    return lambda x: (f(x + dx) - f(x - dx)) / (2*dx)

df = deriv(foo, .001)

def nth_deriv(f, n, dx):
    if n == 0:
        return f
    return nth_deriv(deriv(f, dx), n-1, dx)

x = nth_deriv(foo, 3, 1e-3)(5)
```

```
; Scheme code
(begin

(define (foo x) (* 2 x x x x))

; your code below:









)
```

Question 3a: Rewrite the `sqrt` function without using loops. This function should return the same input as the original version for all integers n <= the recursion limit).

```
def sqrt(x, epsilon):
    guess = x / 2
    while abs(guess ** 2 - x) > epsilon:
        guess = (guess + x/guess) / 2
    return guess
```

```
def sqrt(x, epsilon):
   # your code below
```

Question 3b: Write the `sqrt` function below in Scheme:

R22 Participation Credit **Kerberos : _____@mit.edu**

Hand this sheet in at the end of recitation to get participation credit for today.

Question 1: For each of the four statements written in Python below:
-What is the equivalent expression written in Scheme?
-What will the output of interpreting that expression be?
-How many times will evaluate be called in the course of interpreting that expression?
Note, example A has been completed for you.

```
; Example A:
; x = 4          ; provided Python code
(define x 4)     ; Scheme equivalent
; output: 4
; # calls to evaluate: 2 – why?

; Example B:
; y = x - 1



; output:
; # calls to evaluate:


; Example C:
; square = lambda s: s * s



; output:
; # calls to evaluate:



; Example D:
; z = square(x) + square(y)



; output:
; # calls to evaluate:
```