

MORE SCHEME

COMPUTER SCIENCE MENTORS CS 61A

April 8 to April 10, 2019

1 Scheme

1. What will Scheme output? Draw box-and-pointer diagrams to help determine this.

(a) `(cons (cons 1 nil) (cons 2 (cons (cons 3 (cons 4 nil))
 (cons 6 nil))))`

(b) `(cons (cons (car '(1 2 3)) (list 2 3 4)) (cons 2 nil))`

(c) `(define a 4)
 ((lambda (x y) (+ a)) 1 2)`

(d) `((lambda (x y z) (y x)) 2 / 2)`

(e) `((lambda (x) (x x)) (lambda (y) 4))`

(f) `(define boom1 (/ 1 0))`

(g) `boom1`

(h) `(define boom2 (lambda () (/ 1 0)))`

(i) `(boom2)`

(j) How can we rewrite boom2 without using the lambda operator?

2. What will Scheme output?

- (a) `(if 0 (/ 1 0) 1)`
- (b) `(and 1 #f (/ 1 0))`
- (c) `(and 1 2 3)`
- (d) `(or #f #f 0 #f (/ 1 0))`
- (e) `(or #f #f (/ 1 0) 3 4)`
- (f) `(and (and) (or))`

3. What will Scheme output?

- (a) `(define c 2)`
- (b) `(eval 'c)`
- (c) `'(cons 1 nil)`
- (d) `(eval '(cons 1 nil))`
- (e) `(eval (list 'if '(even? c) 1 2))`

2 Scoping

1. What is the difference between dynamic and lexical scoping?
2. What would this print using lexical scoping? What would it print using dynamic scoping?

```
a = 2
def foo():
    a = 10
    return lambda x: x + a
bar = foo()
bar(10)
```
3. How would you modify an environment diagram to represent dynamic scoping?

3 Code-Writing

1. Define `is-prefix`, which takes in a list `p` and a list `lst` and determines if `p` is a prefix of `lst`. That is, it determines if `lst` starts with all the elements in `p`.

```
; Doctests:
scm> (is-prefix '() '())
#t
scm> (is-prefix '() '(1 2))
#t
scm> (is-prefix '(1) '(1 2))
#t
scm> (is-prefix '(2) '(1 2))
#f
; Note here p is longer than lst
scm> (is-prefix '(1 2) '(1))
#f

(define (is-prefix p lst)
```

```
)
```

2. Define `apply-multiple` which takes in a single argument function `f`, a nonnegative

integer n , and a value x and returns the result of applying f to x a total of n times.

```
;doctests
```

```
scm> (apply-multiple (lambda (x) (* x x)) 3 2)
```

```
256
```

```
scm> (apply-multiple (lambda (x) (+ x 1)) 10 1)
```

```
11
```

```
scm> (apply-multiple (lambda (x) (* 1000 x)) 0 5)
```

```
5
```

```
(define apply-multiple (f n x)
```

```
)
```