

CS 330 – Programming Languages
HW5 Scheme 1

Name Joshua Haupt Grade

Part 1 Warming up - tell the result from evaluating the following scheme expressions. You will use DrRacket (Language = *Pretty Big*) to verify your answers for this assignment. (2x20=40 points)

- | | | |
|---|----|--|
| (a) (cdr (car (cdr '((a b) (c d))))) | => | _____ (d) _____ |
| (b) (car (car (cdr '(cdr ((a b) (c d) e f))))) | => | _____ (a b) _____ |
| (c) (caaddr '((a b) (c d) (e f))) | => | _____ e _____ |
| (d) (car '(car (cdr (cdr ((a b) (c d) e f))))) | => | _____ car _____ |
| (e) '(car '(car (cdr (cdr ((a b) (c d) e f))))) | => | _____ (car '(car (cdr (cdr ((a b) (c d) e f))))) _____ |
| (f) (cons 'a '(c d)) | => | _____ (a c d) _____ |
| (g) (append '(a b) '(c d)) | => | _____ (a b c d) _____ |
| (h) (list '(a b) '(c d)) | => | _____ ((a b) (c d)) _____ |
| (i) (member 'a '(a b c)) | => | _____ (a b c) _____ |
| (j) (list '(b c) (list 'a)) | => | _____ ((b c) (a)) _____ [f with sublists] |
| [Compare and contrast this with f] | | |
| (k) ((lambda x x) 'car 'cdr 'list) | => | _____ (car cdr list) _____ |
| (l) (symbol? 'a) | => | _____ #t _____ |
| (m) (null? '()) | => | _____ #t _____ |
| (n) (reverse '(a (b c) d)) | => | _____ (d (b c) a) _____ |
| (o) (length '(a (b c) (d) e)) | => | _____ 4 _____ |
| (p) (display "Hello World!") | => | _____ Hello World! _____ |
| (q) (Write "Hello World!") | => | _____ "Hello World!" _____ |
| (r) (let ((a 2)) (set! a (read)) a) | => | _____ Hello _____ |
| (input is <i>Hello World!</i>) | | |
| (s) (append '(b c) (list 'a)) | => | _____ (b c a) _____ [reverse of f] |
| [compare with f] | | |
| (t) (reverse (cdr (reverse '(x y z)))) | => | _____ (x y) _____ |

Part 2 Making Procedures - Define the following five scheme procedures and store them in a single scheme file name "*yourlastname1.rkt*". Defining helper procedures may be useful in solving some of these problems. Be sure to include your name as comments at the beginning of the file. (10X4=40 points)

1. `countAtom` – count the number of Atoms in a list. You may assume there is no sub-list. For example,

`(countAtom '(1 "a" df)) → 3`

2. `countAllAtom` – count the number of Atoms in a list including those in sub-list. For example,

`(countAllAtom '(df 2 (3 (4 r) 123 "a")))) → 7`

3. *double7* - a function that takes a list of integers, and return a list with all 7s doubled. You may assume there is no sublist.

For example, `(double7 '(1 2 7 17 72 7 3)) → '(1 2 14 17 72 14 3)`

4. *doubleAll7* - a function that takes a list of integers, and return a list with all 7s doubled, including those in sub-list.

For example, `(doubleAll7 '(1 (2 (7 (17 72 7)) 3))) → '(1 (2 (14 (17 72 14)) 3))`

This assignment is worth 80 points and is due on Monday (10-31) at class time. Please turn in this document at the beginning of class and submit a softcopy of your scheme definitions in a single file on Moodle.