

CS3 – FALL 07: FINAL REVIEW SESSION

1. **count-sublists** (credit here to CS3 reader from Dan Garcia Fall 2003)

Write a procedure that takes a generalized list as an argument and returns how many sublists it contains. EX:

```
(count-sublists '(cs3 is cool)) → 0
(count-sublists '(((cs 3 is) really) ((cool)))) → 4
```

2. **deep-filter**

Write `deep-filter` which takes in a list and a predicate, and returns the filtered list (much like `filter`). It must preserve the structure of the list. EX:

```
(deep-filter even? '(((3)) 2 1 ((2 3 4)))) → ((()) 2 ((2 4)))
```

3. **add-to-roster**

Write a procedure, called `add-to-roster`, that takes three lists — an association list called `current-roster`, a list of new player names, and a list of their numbers, correspondingly. This procedure checks to see if the player is already in the roster, and if he is not, adds the new player, with its number, appropriately to the beginning of the roster. USE TAIL RECURSION and NO HELPER FUNCTIONS.

Example of current roster: `((Daei 10) (Mahdavia 2) (Hashemian 9) (Karimi 8))`

4. passed-tests?

Complete the following without the use of explicit recursion or helper functions.

- A. Write a procedure that takes in three lists as arguments. The first list is a list of procedures, each of which only takes a single argument. The second is a list of arguments for those procedures. The third is a list of expected results. Given these three lists, the procedure returns true if all of your test cases pass, false if any of them fail.

- B. Provide a sample call to the procedure you just wrote, along with the correct return value.

5. best-class (Similar to the GPA lab exercise)

Write a procedure called `best-class` that takes a list as its argument. Each element in this list is also a list that contains two elements – the teacher's name and a list of his students' grades.

`best-class` returns a list that contains the name of the teacher with the highest total of student grades and the highest total of student grades. Do not write any additional helpers of your own.

EX: `(best-class '((Johnson (A B A C NP A)) (Smith (C C D P F A)) (Doe (A A B A A NP))))` → `(Doe 24)`

A helper procedure `grade-conversion` is already written for you. It takes a letter grade as its argument and returns its corresponding numerical value. However, the only valid arguments are A, B, C, D and F. Here is the mapping for points: A=5, B=4, C=3, D=2, F=0.

6. Debug/Evaluate

```
(define (mystery1 lst)
  (map (lambda (elm)
        (cond ((not (list? elm)) elm)
              (else (mystery1 elm))))
       lst))
```

- A. Is there an argument that will make `mystery1` crash and generate an error? If so, what is it?
- B. What does `(mystery1 '(a (b) ((c (d) e ((f)))) g))` return? (Assuming all possible bugs have been fixed.)

```
(define (mystery2 lst)
  (cond ((null? lst) lst)
        ((list? (car lst))
         (append (map mystery2 (car lst))
                 (mystery2 (cdr lst))))
        (else (cons (car lst) (mystery2 (cdr lst))))))
```

- C. Is there an argument that will make `mystery2` crash and generate an error? If so, what is the fix?
- D. What does `(mystery2 '(a (b) ((c (d) e ((f)))) g))` return? (Assuming all possible bugs have been fixed.)

7. black-out

With Southwest, you can receive a free roundtrip after 8 roundtrips. However, these free tickets usually have black out dates. Write a procedure that returns the NON-blackout dates. The procedure `black-out` takes two lists as its argument. The first list indicates the blackout dates in this format: ((January 1 15) (July 4) (February 14) (November 27 28 29)). The second list indicates the days you are willing to fly in each month, in this format: ((January (1 2 3 ... 31)) (February (1 5 17)) (March (1 2 9 16 ... 31)) (April ()) ...) The return value should be a subset of the days you are willing to fly, or in that same format but with blackout dates removed.

```
(black-out '((january (1 9 13 15 18 31)) (february 1 5 17) (march 1 8 17 28
31) (april ()) ... (july (1 3 4 18)) (august (4 8 19)) ...) '((january 1 15)
(july 4) (february 14))) → ((january (9 13 18 31)) (february 1 5 17) (march 1
8 17 28 31) (april ()) ... (july (1 3 18) (august (4 8 19)) ...)
```