CS3 Midterm2 Review Problems (Spring 2005)

More complicated recursion

1. Produce a function to find the length of the longest decreasing substring in a sentence of positive numbers. For example:
   `(2 1 5 3 2 1 12)` → `4`
   `(13 12 11 15 10 5 2 1)` → `5`

2. Write a procedure that returns all words representing subsets of the letters in the input sentence. Assume that the input sentence is not empty and includes distinct letters. Note that the order does not matter in out put subsets so `ab` is considered the same subset as `ba`. Example:
   `(a b c)` → `("", a, b, c, ab, ac, bc, abc)`
   `(w e)` → `("", w, e, we)`

3. I have c chips and d drinks, how many ways can I finish all these snack if I ate one at a time? Example: `(snack 1 2)` → `3`, which includes (chip, drink, drink), (drink, chip, drink), and (drink, drink, chip).

4. Given a sentence of words, I want to keep only the words that also have their reversed counterparts in the same sentence, and I want these words to be returned in increasing order of their length. If two words have the same length, then return them in the same order as the original sentence. For example, the argument `'(abc gfed mlkj hi ih defg mlkj mlkj jklm)` should return (`hi ih gfed mlkj defg mlkj jklm)`. You may assume the procedure `reverse` is already written for you: `(reverse 'abc)` → `cba`

Higher-Order-Functions

5. Write function that finds the mode (or one of the modes in case of ties) of the numbers in the input sentence. Example:
   `(1 2 3 12 1 1)` → `1`
   `(12 3 5 3 12)` → `12` or `3`

6. Consider a database of student information that consists of a word for each student of with the following structure: `Name;Age;Letter-Grade`. Assuming that student name includes only English letters, write a procedure that returns a sentence of all student ages. For instance:
   `(Mary;19;A+ Bill;23;B- Jimmy;44;A Sally;25;C+)` → `(19 23 44 25)`

7. Modify your procedure so that it returns the name of (one of) the oldest student(s) in class. So for the previous example the procedure should return: `Jimmy`.

8. Write the procedure `last-out` using higher-order-functions. Here's a sample call:
   `(last-out 'clint)` → `(clint lint int nt t nt int lint clint)`

9. Before you had written in lab `thoroughly-reversed` using recursion. Lets try it with HOF now. `(thoroughly-reversed '(scheme is so cool))` → `(looc os si emehcs)`

10. Write the procedure `match-all` so that every person in the first argument is matched to every person in the second argument. However, if the person's name consists only of numbers, then we want to ignore him/her. Example:
    ```
    (match-all '(mary 123 helen abby) '(ted 456 bob))
          → (mary-ted helen-ted abby-ted mary-bob helen-bob abby-bob)
    ```
    Note the order of the return value: we match every person in arg1 to one person in arg2 before matching them to the next person in arg2. Each match is created with a hyphen between the two names.

Lambda

11. What does the following evaluate to? If it returns an error, describe what the error is and why it occurred. If it returns a procedure, describe what argument(s) it takes and what it does with it/them. If it returns a value, write out the arithmetic expression of it.
    ```
    STk> ((lambda (x y)
            (lambda (a b)
                (+ a ((lambda (g h) (* g h)) b 10) x y))) 5 8)
    ```

12. If the previous expression generated an error, fix it and make a procedure call. If it returned a procedure, how do you call that procedure with arguments `10` and `7` without changing the existing code, and what does it return? (don't add anything inside the left-most parenthesis and the right-most parenthesis.)

13. Assume that I have a function `Even-Apply` that takes two arguments: a one-argument function and a sentence. `Even-Apply` applies the function on all even-position members of the sentence. For example, `(even-Apply butfirst '(hello world! What a happy day))` → `(hello orld! What "" happy ay)`. Write a function that increments all even-position members of a sentence by one. Example: `(1 2 3 4)` → `(1 3 4 4)`, and `(1 1 1 1 1)` → `(1 2 1 2 1)`

14. Use `Even-Apply` to write a procedure that takes a list of student names and their grades and creates a list of student names and their pass or fail situation. Your procedure should take the minimum passing grade as an input. See the example for more clarification:
    Passing grade = 60 => `(Ali 75 John 55 Maria 90 Edward 85 George 40)` → `(Ali passed John Failed Maria Passed Edward Passed George Failed)`

15. Write a function that categorizes all names in a sentence based on their first letter. It should put all names starting with A first; then all names starting with B; and so on… For instance if the input sentence is `(Adam Bobby John Catherine Sara Bill Abraham Jim)`, the function will return `(Adam Abraham Booby Bill Catherine John Jim Sara).` [this is the first step to implement bucket sort!]

Others: Analyze/Debug

16. There is a monkey named Sam. It will skip and hop for you depending on how many coins you give it. We have written a procedure `monkey` that will return a sentence describing the order in which Sam skips and hops when given n coins. We have also written the procedures `count-skip` and `count-hop` to count the number of times Sam skips and hops, respectively, when given n coins. Example: `(count-skip (monkey 5))` → `7`

```
(define (monkey coin)
   (cond ((= coin 0) '())
         ((= coin 1) '())
         (else (se 'skip (monkey (- coin 2))
                   'hop (monkey (- coin 1))))))
```

a) What does `(monkey 4)` return?
b) What does `(count-hop (monkey 3))` return?
c) Do `count-hop` and `count-skip` always return the same value when given the same argument? Why?
d) Write a formula for estimating `(count-skip (monkey n)).` What is the pattern similar to?

17. You just finished writing the procedure `condense`, but the neighbor's kid came to play on your computer before you could save it. The correct procedure should do the following: `(condense '(1 2 3 a b 8 9))` → `(6 ab 17)` Here's the buggy code:

```
(define (condense sent)
  (cond ((empty? sent) sent)
        ((and (number? (first sent))
              (word? (second sent)))
         (se (first sent) (condense (bf sent))))
        ((and (number? (first sent))
              (number? (second sent)))
         (condense (se (+ (first sent) (second sent)) (bf (bf sent)))))
        ((number? (second sent))
         (se (first sent) (condense (bf sent))))
        (else (condense (se (word (first sent) (second sent))
                            (bf (bf sent)))))))
```

With the buggy code, what does this return: `(condense '(1 2 3 a b 8 9))`? Explain why. Fix and explain all the bugs in the code so that it returns the correct value.