UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

CS 263 **Spring 1994**
K. A. Yelick

CS 263: Design of Programming Languages
Homework 1
Due February 6, 1996

1. Do exercise 2.2.4 from Mitchell's book.

2. Do exercise 2.2.15 from Mitchell's book.

3. Do exercise 2.3 from Mitchell's book.

4. There is no facility in Lisp to explicitly create and destroy memory. Instead, implementations of Lisp do garbage collection to find and recover storage that is no longer accessible from the program. Consider the problem of designing a garbage collector for pure Lisp, which has lambda abstraction to define functions, functional application, `cons` to create pairs of things and `car` and `cdr` to take them apart, as well as integers and booleans plus some basic functions. Whenever a `cons` appears in a Lisp problem, a new pair is allocated on the heap. The problem is to determine when these can be reclaimed.

A simple technique for implementing garbage collection is to associate a count with each object (in our case with each `cons` cell). When a new cell is allocated, the count is set to 0. When a pointer is set to point to the cell, the count is incremented by 1. If a pointer is reset or destroyed, the count of the cell it used to point to is decremented. When a count reaches 0, the cell is considered garbage and is added to a list of free storage.

a. Describe how reference counting could be used for garbage collection in the following expression:

```
(car (cdr (cons (cons a b) (cons c d))))
```

Assume the reference counts for `a`, `b`, `c`, and `d` are initially 2. Which of the `cons` cells in the above expression will be reclaimed after the expression has been evaluated?

b. Why does the addition of `rplaca` or `rplacd` to Lisp make reference counting unusable? For those not familiar with Lisp, `rplaca` is used to reset the first field of a pair and `rplacd` is used to reset the second field. For example, `(rplaca (cons 4 6 ) 3)` produces the pair containing `3` and `6`, while `(rplacd (cons 4 6 ) 3)` produces the pair containing `4` and `3`.