CS 3 Midterm #2 — April 5, 1994

Your name _____

login cs3–_____

Discussion section number _____

TA's name _____

This exam is worth 15 points, or about 14% of your total course grade. The exam contains four questions.

This booklet contains five numbered pages including the cover page. Put all answers on these pages, please; don't hand in stray pieces of paper. This is an open book exam.

When writing functions, write straightforward code. Do not try to make your program slightly more efficient at the cost of making it impossible to read and understand.

When writing procedures, don't put in error checks. Assume that you will be given arguments of the correct type.

Our expectation is that many of you will not complete one or two of these questions. If you find one question especially difficult, leave it for later; start with the ones you find easier.

• • • **You may NOT use higher-order functions in your answers! Use recursion instead.**

1	/2
2	/4
3	/4
4	/5
total	/15

Question 1 (2 points):

What is the value of each of the following expressions? (If the expression's value is a procedure you may just write "procedure." If evaluating the expression would cause an error, just write "error.")

```
> (map (lambda (x) (+ 3 (cadr x))) '((1 2 3) (4 5 6) (7 8 9)))
```

```
> (append 'for '(no one))
```

```
> ((lambda (a) (cons (cadr a) a)) '(w x y z))
```

```
> (first (bf (cadr (cadr '((john lennon) (paul mccartney)
                           (george harrison) (ringo starr))))))
```

Your name _____ login cs3–_____

Question 2 (4 points):

• • • **You may NOT use higher-order functions in your answers! Use recursion instead.**

In the game of Geography, each player must name a state or city or country whose first letter is equal to the last letter of the place that the previous player named. For example, if one player says "Berkeley" then the next player can say "Yucatan" and the one after that can say "Norway."

Write a predicate procedure `geography?` that takes a sentence as its argument and returns `#t` if the words of the argument form a valid sequence of Geography names. (You only have to worry about the last and first letters matching, not about whether the words are real place names!) For example:

```
> (geography? '(berkeley yucatan norway yonkers somerville england))
#t
> (geography? '(berkeley yonkers norway yucatan))
#f
```

Question 3 (4 points):

• • • You may NOT use higher-order functions in your answers! Use recursion instead.

Write a procedure `maprest` that takes two arguments, a function and a list, like `map` except that instead of applying the function to each element of the list, `maprest` should apply the function to the *subset* of the list from each element to the end. For example,

```
(map count '(a bunch of words))
```

means

```
(list (count 'a)
      (count 'bunch)
      (count 'of)
      (count 'words))
```

so the result is `(1 5 2 5)`, but

```
(maprest count '(a bunch of words))
```

means

```
(list (count '(a bunch of words))
      (count '(bunch of words))
      (count '(of words))
      (count '(words)))
```

so the result is `(4 3 2 1)`. Another example:

```
> (maprest reverse '(tell me why))
((why me tell) (why me) (why))
```

Your name _____ login cs3–_____

Question 4 (5 points):

• • • **You may NOT use higher-order functions in your answers! Use recursion instead.**

Write a procedure `smallest-sum` whose argument is a tree of numbers. It should return the smallest possible sum of a chain of numbers from the root to some leaf of the tree. For example, given the tree

your procedure should return 21, which is $10 + 9 + 2$.

The argument uses the tree abstract data type, not a "cheap tree."

Notice that the smallest sum in the example above is *not* found by choosing the smallest child of the root! Instead you must choose the subtree with the smallest `smallest-sum`.