Com Sci 221 — Programming Languages
Final Exam — Winter 1989
Wednesday, March 15
2 pages, 7 problems
Open book, open notes, open mind

Each numbered problem is equally valuable.

1. Give 3 examples of features that prevent a local variable in a recursive procedure from being allocated on the stack. For each such feature, give an example of a program using that feature in which stack allocation would fail.

2. For each of the following problems decide whether mark–and–sweep (the "lazy approach" to deallocation on p. 290 of the text) or reference–count (the corresponding "eager approach") is likely to be more satisfactory, assuming that time is a more critical resource than space. Explain each choice briefly. Concentrate on three issues: interactive response time vs. total time, time proportional to the amount of space used vs. time proportional to the amount of space recovered, and time spent when storage is allocated vs. time spent when storage is deallocated. For the purposes of this problem, there is no point in considering copying collectors, as their behavior is qualitatively the same as mark–and–sweep.

   (a) A desk–calculator program is required to evaluate each arithmetic expression typed in by the user interactively, producing the value of one expression before the user types the next.

   (b) A simple function must be applied to each integer in a long list, producing a list of results. The result list must be allocated, but little or no deallocation is expected.

   (c) A database system is implemented by storing all data in a huge Lisp S–expression called **data**. This S–expression changes very infrequently, but many queries are processed every day in batch mode. Queries are expressed as Lisp functions that are automatically applied to **data**.

3. Translate the arithmetic expression $2*(x+y)*(x+1)$ into postfix form, assuming that $*$ is left–associative. Explain briefly the relationship between the postfix expression and the code that would be compiled for the expression. What common operators require code that does not correspond nicely to postfix expressions, and why?

4. What type will ML deduce for each of the objects defined in the following fragments?

   fun identity x = x;
   fun repeat f x = f (f x);
   val stutter = repeat identity;
   val quadruplicate = repeat repeat;
   fun compose f g x = f (g x);
   fun duplicate f = repeat (compose f) identity;

5. Do problem 4.3 on p. 162 of the text.

6. What stream is the value of **mystery** in the following Scheme definitions?

```
(define (add-streams s1 s2)
        (cons-stream (+ (head s1) (head s2))
                (add-streams (tail s1) (tail s2))))
(define mystery
        (cons-stream 1 (cons-stream 1
                (add-streams mystery (tail mystery)))))
```

7. Write an iterative (i.e., **while**–loop) program fragment to do binary search on an array of 100 integers that has been sorted into increasing order. The method is outlined on p. 74 of the text, in Example 3.2. Annotate your program fragment with appropriate pre–assertion, post–assertion and loop invariant to express the essentials of its behavior. Hint: try to express all of the assertions in terms of the arithmetic inequalities $\leq$ and/or $\geq$. By doing so, you will simplify their forms, and avoid confusing conditions that depend on whether the element searched for is actually present or not.