Com Sci 221 — Programming Languages
Assignment #8 — Winter 1994
Due Friday, 18 March 10:30 AM
Deliver to instructor at or before final exam

1. Use stream operations in Scheme to compute the stream *hamming* described in problem 9.5 on p. 376 of the text. The stream operations that you need are `cons-stream`, `stream-car`, and `stream-cdr` (these are called `qcons`, `head`, and `tail` in the text). Discuss the difference between your stream–oriented program for the Hamming sequence and the sort of program that you would have written in a language without streams.

2. Do problem 7.3 on p. 292 of the text as a pure functional program in ML. Do part (b) first, then apply it to solve part (a), and also to sort lists of words into alphabetical order.

3. A word x precedes a word y in *anagram order* if, after sorting the letters in x and y into alphabetical order, the resulting scrambled version of x precedes the scrambled version of y. For example, "zebra" precedes "accordion", because the scrambled form of "zebra" is "aberz", and the scrambled form of "accordion" is "accdinor". Notice that anagrams are equal in this ordering — for example "pots", "stop", and "tops" all have the scrambled form "opst". Build a program to sort lists of words in anagram order, using the solution to the previous problem for the sorting of letters within a word, and also for the sorting of scrambled words. The final result, however, should contain the *original* words from the input in anagram order, *not* the scrambled forms. Comment on the difference between your ML solution and the sort of solution that you would use in a language without functions as first-class objects. Comment also on the types of the functions you defined.

4. Define interesting control structures as higher-order functions in ML. Include at least

   (a) **While**-loops in the style of Pascal,

   (b) **For**-loops in the style of C,

   (c) **Case**-commands in the style of Pascal.

   For extra credit, add more sophisticated control structures. For ideas, look at the treatment of exceptions in ML. The goal is not to get a precise implementation of C or Pascal, but rather to reproduce the essential utility of their control structures within ML. You do not need to run test cases for these functions, but you should give the definitions to the ML interpreter in order to see what types are assigned. Discuss the differences between the ML functions and the procedural-language control structures that they imitate.