CS 505
Autumn 1997
Assignment 2 — Continuations; ML
Due Oct 17

1. The purpose of this question is to do a bit more exploration of continuations. The Scheme program `~borning/505/factorial.s` on orcas contains a more-or-less normal recursive version of factorial, augmented with some write statements to print out information during the computation. Copy this file to your directory and load it using `(load "factorial.s")`. Evaluate `(factorial 7)` to make sure it's working. The function `continue-factorial` is similar, but captures two continuations in the global variables `continue1` and `continue2`. Evaluate `(continue-factorial 7)`. Now evaluate `(continue1 'whatever)` and then `(continue2 'whatever)`. Explain what happened.

2. As a warmup for ML, write the `range` function in ML. `range` should take a tuple with a pair of numbers: a start and a stop. It should return the list of integers between `start` and `stop` inclusive.

3. Tail recursion is just as relevant a concept to ML as it is to Scheme. If your ML `range` function for Question 2 isn't tail recursive, define another version that is.

4. Assignment 1 included a `sum-tree` function in Scheme. For example, `(sum-tree '(1 2 ((3) 4)))` would evaluate to 10.

   You can't write a similar function `sum_tree` in ML, in other words, `sum_tree [1,2,[[3],4]]` isn't permitted. Why not?

   Write and test an appropriate ML tree type for binary trees of integers, and a `sum_tree` function that does work correctly. Instances of your integer tree type should hold an integer at the leaves; non-leaf nodes should hold a left and right child. Do something appropriate about empty trees (either don't allow them or return 0).

5. Now define a polymorphic tree type that can be used to represent binary trees of any type. Write a polymorphic `tree_reduce` function, analogous to the Scheme function you wrote for Assignment 1. Use `tree_reduce` to define `sum_tree`, and also `product_tree`, `concat_tree`, `and_tree`, and `or_tree`, which find the product, string concatenation, logical and, and logical or of the leaves in a tree. (You will want to define your own functions `my_and` and `my_or`; the builtin operators `andalso` and `orelse` don't seem to be real functions.)

   What is the type of `tree_reduce`? Describe how ML deduces this type.