Department of Computer Science and Engineering
University of California at San Diego

CSE 230
Winter 1998

REVISED Problem Set 3

DUE TUESDAY JANUARY 27 IN CLASS.

1. Consider an infix type operator $+$ that returns a disjoint sum type.

   (i) What feature of standard ML gives about the same effect as this operator?

   (ii) Nevertheless, suppose the $+$ type operator were added to ML, and suppose there were builtin polymorphic constructors `inleft`: $\alpha \to \alpha + \beta$ and `inright`: $\beta \to \alpha + \beta$, and destructors `outleft`: $\alpha + \beta \to \alpha$ and `outright`: $\alpha + \beta \to \alpha$.

   Explain why some expressions using these operators cannot be type-checked at compile-time. Is this compatible with the spirit of ML? Is it possible for ML to deal sensibly with these polymorphic constructors and destructors?

   (iii) Suggest a *single* builtin polymorphic function that would provide the functionality of `outleft` and `outright` and would allow full compile-time type-checking. Be sure to specify the signature of your suggested function clearly.

2. (i) Define your own list append function in ML, and use it in the obvious function to flatten binary trees. What is the time complexity of this function?

   (ii) Define a less obvious function that flattens binary trees in linear time.

   Your function for part (i) is in essence a formal specification, and the function for part (ii) is an efficient implementation of this specification. As part of the next homework assignment, you will use structural induction to prove that your two functions are equivalent, i.e. that for the same input they always yield the same output. In order to make this proof easier, make sure that both functions are as straightforward as possible, and written as cleanly as possible.