Locally Defined Procedures and Sorting

9-20-2004

Opening Discussion

- Do you have any questions about the quiz?
- What did we talk about last class?
- Who can remember some different sorting algorithms that you worked on in different courses? How does each of those sorting algorithms work?

Letrec: a let for Recursion

- One problem with let is that the expressions defining the values of a var can't refer to any of the variables in the let expression. One thing this does is completely eliminate recursion since a recursive function must refer to itself by name.
- For this purpose, Scheme has letrec. It looks the same and let and does basically the same thing, but now expressions in it can refer to names created in it.

Using letrec

- One use for letrec is for when you need to wrap helper functions like we saw for reverse and Fibonacci. The helper functions don't really belong in the global namespace so letrec is better than define.
- Let's rewrite write reverse-all, which does a deep reverse, but with the use of a helper function sitting in a letrec.

Sorting in Scheme

- We can sort lists in Scheme just as we could arrays in other languages. However, some algorithms are better than others for lists.
- Let's write insertion sort in Scheme. This sort is nicely adapted for lists where inserting elements is fairly easy.
- Now let's see what we can do about writing mergesort and quicksort, two sorts that we would typically write recursively even in an imperative language.

Minute Essay

- How do you think these sorts we did today compare to ones you might write in an imperative language? Think in terms of operations performed, not overall performance. Do they make sense in Scheme?