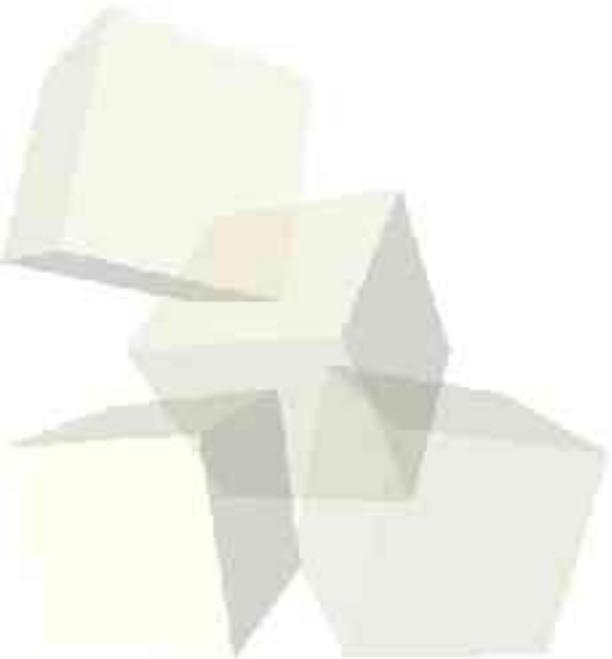
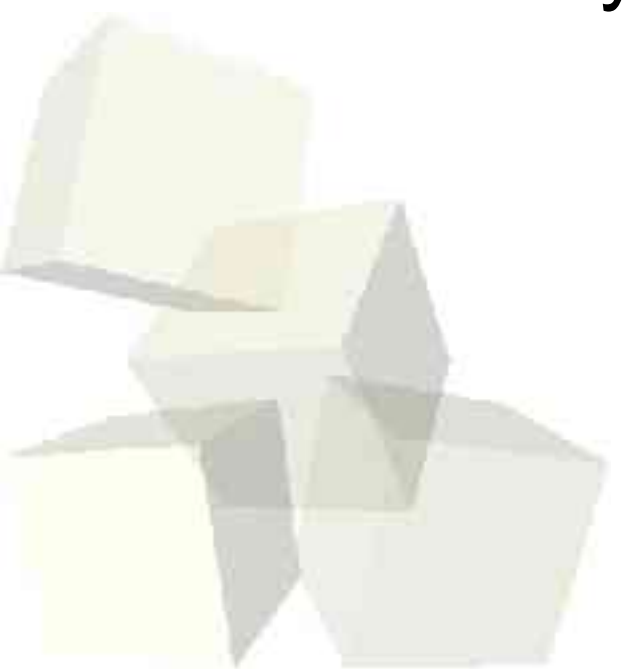
Locally Defined Procedures

9-17-2004



Opening Discussion

- What did we talk about last class?
- Lists have to be walked. You can't do random access. Arrays/vectors allow fast random access. Can they be done well immutably?



Funky Shortcut

- For some reason you book doesn't mention the eval function. The way Scheme (and LISP) works is that every time you type something in, it calls eval on it. All a quote does is tell eval not to evaluate something.
- So if you have a list that you want to apply some function to the elements of you can cons the function to the list and pass it to eval.
  - (eval (cons '+ '(2 3 4 5)))

Using eval to Turn Lists into Functions

- One of my biggest complaints about a language is when it leads you to think a thought, but won't let you implement that thought. I have had that happen some in C++ and I'm running into that a lot in Scheme.
- After significant thought I figured out how to turn a list into a function: call eval on the EXACT list that represents the function. So a list starting with lambda.

Local Names and Let

- As you might have noticed, define only works at the global level and once a name is bound, it can't be reused.
- Sometimes it is nice to have a local name to refer to something by. This can help organize expressions or is really useful when an expression would have to be done multiple times.
- The syntax for let is as follows:
  - (let ((var1 val1) (var2 val2) ...) expr)
- It takes the value of the expression.

Code

- We want to spend the rest of the day coding to look at what we have learned so far.
- First let's do another deep recursion: reverse-all.
- Let's write a DT to do some inventory work. It's not exciting, but it will probably help those who haven't finished the assignment.
- Other things we could play with today are writing a N-body simulator or symbolic differentiation. They seem more fun to me.

Minute Essay

- Write a function that takes one list as an argument and splits it in two, returning a list with two lists in it. I don't care how you split the elements, just so half go in one list and the other half go in the other.
- Remember that we have quiz #2 on Monday.