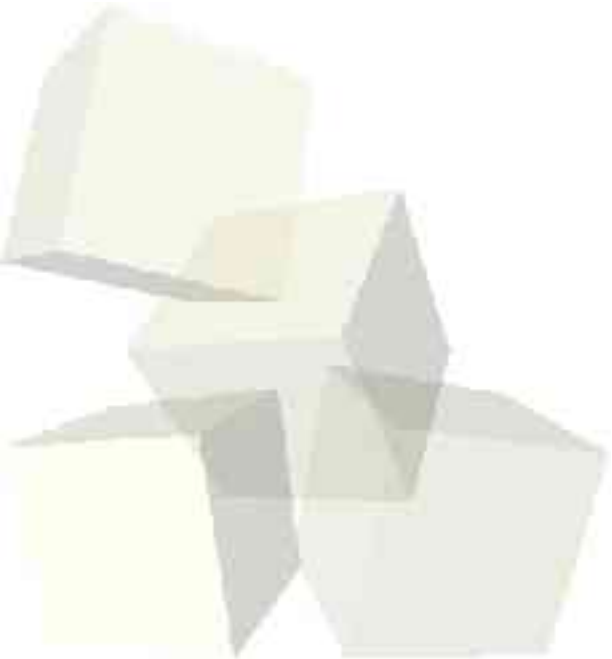
Abstracting Procedures

9-27-2004



Opening Discussion

- Last time we didn't introduce anything new, instead we spent time coding. What were the problems we worked on coding? Can someone review the approach to them?
- Today we move onto a new topic that really uses the functional nature of the language: abstracting procedures/functions. I showed you a tiny sampling of this at one time in the past with a function that we passed another function to. Does anyone remember what that was?

Functions as Arguments

- On of the great powers of functional languages is that we can use functions just like anything else. That includes the ability to pass functions as arguments to other functions.
- We did this early in the semester in our map function which applied a function to every element of a list and returned a list of the results.
- A similar function is for-each that works on functions that don't return values like display.

Passing Functions to Sorts

- One way in which you could use the ability to pass in functions is to make a sort polymorphic.
- This was mentioned before, but this is the way to implement it.
- Make one of the arguments to the sort be a function that serves as a comparator. Depending on the nature of the sort, a simple predicate for less? could work or you might want a full comparator that returns an int.

Variable Argument Counts

- Sometimes we would like to be able to call a functions with a variable number of arguments like they do with + and min.
- You can do this with a lambda expression that doesn't have the arguments in parentheses.
  - (lambda var expr1 expr2 ...)
- In this form, var is a list of the arguments that are passed in. You can then walk this list to do what you want with the arguments. This uses the implicit begin.

Functions as Return Types

- Just as we can pass functions into functions, we can also have functions return functions.
- A simple mathematical example if this is function composition which takes two functions and returns a function that is their composition.
- (define (compose f g) (lambda (x) (f (g x))))
- We'll see a somewhat different example of this type of behavior in a few slides.

Ackermann and Math Functions as a Hierarchy

- You know from grade school that multiplication is repeated addition and exponentiation is repeated multiplication.
- We don't give names to them, but one can obviously add higher order math functions to this hierarchy.
- We can write a function that takes one argument and returns a function of two values that does the math operation of the proper level.
- The Ackermann function is the nth level function applied to n and n.

Currying

- So far when we have had functions of multiple variables, we have just written them as that. There is another approach. Instead of writing a function of two variables, we can write a function of one variable that returns a function of one variable.
- This can obviously be extended to more arguments as well.
- This could be useful in your assignment because you can write an add-grade function that is curried that takes a grade first and a student later after it is found.

Minute Essay

- How close can you come to doing any of the things we talked about today in other languages? What is the language and how close can you come to what part of our topic for today?
- Most languages give you the ability to do something similar to one or two of the topics from today. The real question is do you realize it and how does it compare.
- Remember that assignment #3 is due today. Quiz #3 is Wednesday.