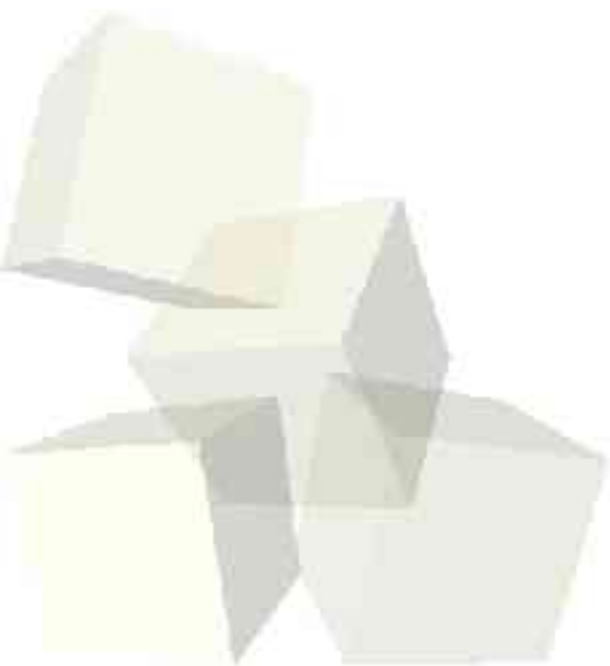




Recursion, Patterns, and Let

10-20-2004





Opening Discussion

- Who can describe the data types that we discussed in ML last time? How do we write functions in ML?
- Functions for cube and “doubling” a string. Remember the operators and other details of ML.





References to External “Variables”

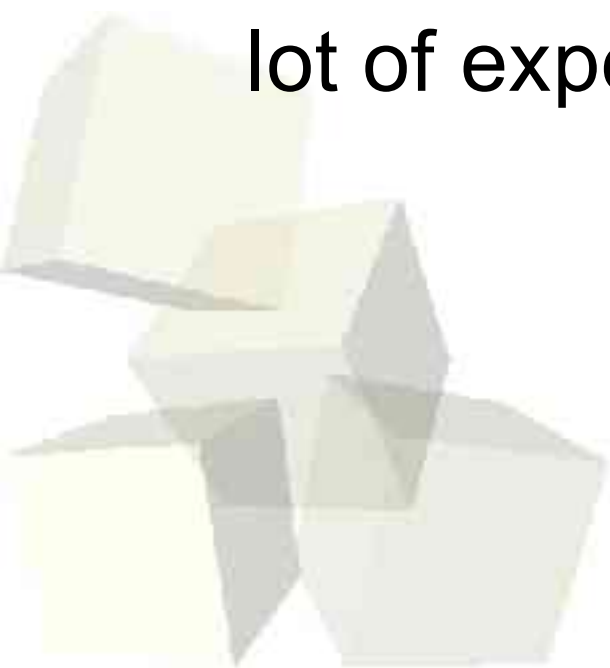
- In ML, if you define a function that refers to an outside value name, it uses the value bound to the name at the time the function is defined.
- Note that this means the order in which things are defined can be important. Things must be defined before you can use them, like in C.





Recursion

- Recursion in ML is done just like in Scheme and because they are both functional it is used a lot.
- I'm not going to specifically focus on recursion though because you now have a lot of experience with it from Scheme.





Mutual Recursion

- The idea that something must be defined before you can use it causes problems for mutual recursion. That is when A calls B and B calls A.
- In C you use function signatures to declare things before you define them.
- In ML we can use a keyword “and” along with fun.
 - ◆ fun <def1>
 - ◆ and <def2>
 - ◆ ...
 - ◆ and <defN>;



Patterns

- At the end of last class I showed you the concept of a pattern in ML. A function can have multiple different definitions where each definition depends on matching a certain pattern in the argument(s).
- A pattern is either a constant or has some type of structure. It can't be a boolean expression though.
- The multiple definitions are separated by “|”.
- It is possible that the patterns don't match all possibilities which gives a warning.



“as” in Patterns

- We saw how a pattern can be a list. Sometimes it is nice to have a value in two different ways.
- We can do this with “as” so in place of the pattern we put <identifier> as <pattern>.
- So now the argument as a whole goes by the first identifier or we can use the pattern pieces for it.
- Let's look at how this works using a merge function.



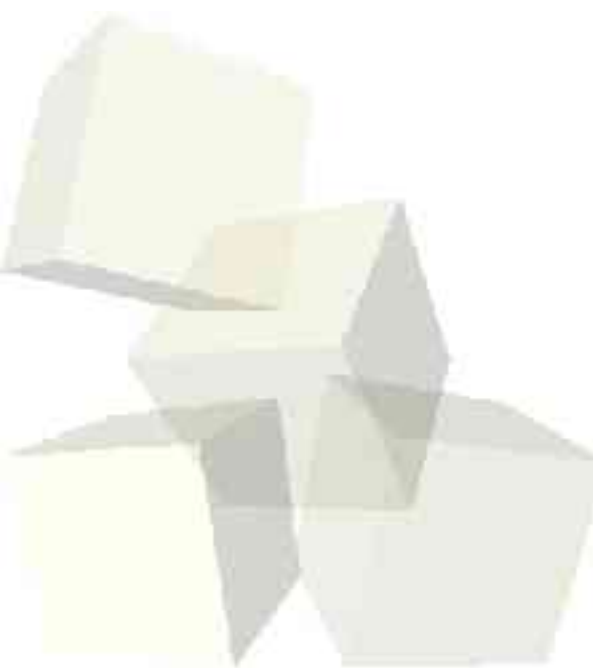
Anonymous Variables and Pattern Details

- You can also use a “_” to specify a part of a pattern that you don't care about. You do this if some argument isn't needed for a particular implementation.
- You can't repeat the same identifier twice in a single pattern.
- Patterns can include parentheses to group sub-tuples and sublists. Note that even for sublists you use parentheses because you are just setting order of operation.
- Can't use @, math ops, or reals in patterns.



Let


- Just like Scheme, it is frequently helpful to be able to bind names to values in a function in ML and we can do this with let.
- Note that the semicolons can optionally follow vals.



```
let
  val <var1>=<exp1>
  val <var2>=<exp2>
  ...
in
  <expression>
end
```



Patterns in val

- When a val gets the value of a function call, we can also use patterns on the left side of a val statement to take apart tuples or lists.
 - This is most useful if a function returns a tuple. To illustrate this, let's write split and mergeSort functions.
 - Here again we see how patterns prevent us from using hd and tl or even using the #i syntax for getting parts of tuples.
- 



Minute Essay

- Using patterns, write a function that returns the length of a list.
- Remember that assignment #5 is due on Friday.

