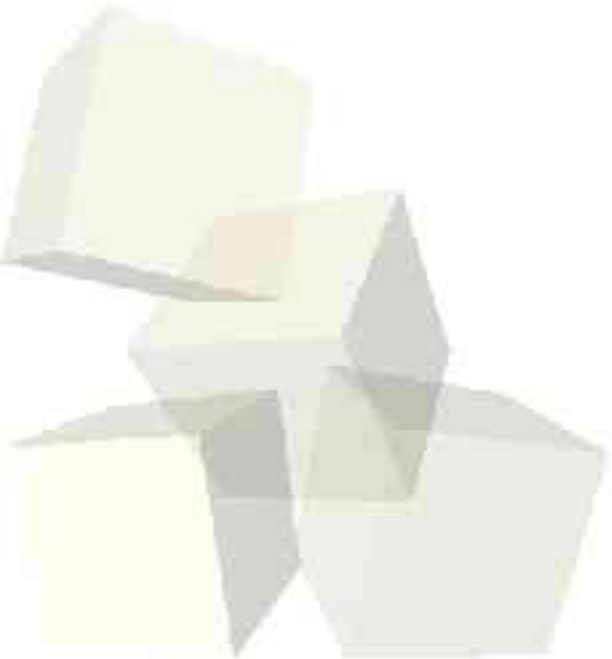
Records, Arrays, and References

11-3-2004



Opening Discussion

- What did we talk about last class? Let's go look at code for the one function we didn't get to last time and test the functions.
- Do you have any questions about assignment #7? Let's talk about the “long term” options.
- Given what you know so far, how would you implement something like a struct from C in ML? Is this approach satisfying? What could be better about it?

Record Structures

- ML has another way of grouping information called a record structure. It turns out that tuples are a special form of this with a special syntax.
- We denote a record structure by putting labeled values inside curly braces.
  - val myItem={name=”shoes”,quantity=5,price=3.0};
- The type of a record has basically the same structure with colons instead of equals and the type on the right.
- Order of the elements doesn't matter.

Extracting Field Values

- Here we see that tuples are just records by the similarity in syntax.
- To get an field value out of a record we do #<label>(<record>).
- So in the example on the last slide we could use #name(myItem) to get the name of the item.
- Tuples are just a special form of records where the labels are consecutive numbers starting at 1. So (v1,v2,v3) is short for {1=v1,2=v2,3=v3}.

Records and Patterns

- As one would expect, we can make patterns that match records. These patterns have a comma separated list of <label>=<pattern>, where <label> is the name of a field in the record.
- You can leave some fields unspecified by putting an ellipsis (...). However, ML must be able to determine the whole type at some point. This is another case where you might use type and specify the type of an argument.

More Records and Patterns

- An example of the ellipsis would be {name=n,...} if all you are interested in is the name. Something else in the function must provide the full type though.
- Even more shorthand, you can leave out the = and a pattern is you are good with having the label name be the local name.
- So {name,...} would pull off the name from the record and let you refer to it locally as just name.

Arrays

- ML provides a mutable array data type that we can use in applications where that will speed up processing.
- Functions for arrays are in the structure Array.
- Array indices start at 0.
- Arrays.array(n,v) returns an array with n elements all set to the value v.
- Array.sub(A,i) gets the ith element of A.
- Arrays.update(A,i,v) sets the value of the ith location in A to v.

Reference Types

- You can also create single elements are are mutable by making them reference types.
- We declare a reference type by prepending the value with the keyword ref.
  - val i = ref 0;
- To get a value out of a reference we have to use the operator ! As a prefix operator.
  - !i=0 is true
- We can assign the value of a reference with :=.
  - i := 2 makes !i=2 be true

While-Do Statement

- ML does provide a statement for doing loops with references. This should only be done if it speeds things up or significantly simplifies code.
  - while <expression> do <expression>
- Of course, this only makes sense if the second expression modifies a mutable value in the first first expression.
- The value produced by a while loop is always unit. This just furthers the idea that it is not a functional construct.

Minute Essay

- Which of the options for the last four assignments do you think you are most likely to do?
- I'll be gone Friday through next Wednesday so you have no class for a week. You will need to be working on assignment #7 though. Don't try to do that with a few big functions. Break the problem up a lot to make it tractable.