Me

- Eric Mullen
- (the one with the nails)
- One of the two section leaders
- First year PhD student, doing research in programming languages
- I've never been a TA for *this* PL course before, but...
Type Synonyms

What if I want to call an `int * int * int` a date?

```haskell
  type date = int * int * int
```
Generic Types, and Type Generality

What is this 'a thing anyway?

It can stand for any type, but only one type. It is often referred to as a type variable.

You've seen lists, which can hold ints, or strings, but can't hold both at the same time.

Lists have type 'a list
Type Generality, continued

What does it mean for one type to be *more general* than another?

Type $t_1$ is more general than $t_2$ if you can consistently replace type variables in $t_1$ with types (or type variables) to get $t_2$. 
Which is more general?

('a,'b) -> 'a
('a,int) -> 'a

int list -> int
'a list -> int

('a,'b,'b) -> 'a
(int,'b,'c) -> int
Equality types

What if I want to write code that works over any type, but I want to be able to compare whether an 'a is equal to another 'a?
Function Patterns

These are just syntactic sugar, not a fundamentally new concept.

```haskell
fun isEmpty(l) =
    case l of
        [] => true
    | x::xs => false

fun isEmpty([]) = true
    | isEmpty(x::xs) = false
```
Is if/then/else really necessary?

Suppose I tell you there's a bug in the SML compiler, and you can't use if/then/else any more. Also, I want all of your code checked in and working yesterday. What do you do?