A few announcements...

1. hw1, 2, 3 graded & returned (for hw1,2 re-grade, see Shen; for 3, see Elizabeth)

2. general points-off:

   - helper functions should be as local as possible - put them in let-bindings!

   - **NAME your FUNCTIONS** according to write-up (unless not specified), next time there will be point deductions

   - **NAME your turn-in FILE** properly. If not particularly specified by write-up, just name them *hw#.sml*. Future deductions for horrible filenames (ex: blah2.txt)

3. Keep receipts of your turn-in for future reference

Questions?

Think: why would an expression like ”3.0=1.0” generate error?

Equality Types and Equality Operators...huh?

What are the *equality operators*? $=$ and $<>$

What are the *equality types*? int, bool, char, string (but not real), and more - products or lists of equality types

Equality types are the types defined by ML that allow equality to be tested among values of that type.

Example:

```
    val x = (1, 2);
    val y = (2, 3);
```

Evaluate `x=y` and we get...?

Evaluate `x<>y` and we get...?

What about

```
    fun f(x) = x*2;
```

Evaluate `f = f` and we get...?

Evaluate `f = (fn x => x*2)` and we get...?

Boolean Expressions

What logical operators do we have? What arguments do they take?

This is a boolean expression:

```
    Not x And (true Or x)
```

What happens when we bind x to $true$? And to $false$?

How can we express Boolean expressions using datatypes?

```
datatype 'a expr =
Const of bool |
Var of 'a |
Not of 'a expr |
And of 'a expr * 'a expr |
Or of 'a expr * 'a expr
exception UnboundVar
```

Q: come up with some example of a string expression.

Q: a function **eval** that takes a constant expression and returns its value.

```
datatype 'a expr =
Const of bool |
Var of 'a |
Not of 'a expr |
And of 'a expr * 'a expr |
Or of 'a expr * 'a expr
exception UnboundVar

fun eval (Const b) = b
  | eval (Var v) = raise UnboundVar
  | eval (Not e) = not (eval e)
  | eval (And(e,f)) = (eval e) andalso (eval f)
  | eval (Or(e,f)) = (eval e) orelse (eval f)
```

Q: what would `Or(Const true, Var "x")` evaluate to?
Q: what about `Or(Var "x", Const true)`?