Booleans
 Adding Let
 Conclusion
 Booleans
 Adding Let
 Conclusion

 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○

# Interpreters, Part 2

#### Dr. Mattox Beckman

University of Illinois at Urbana-Champaign Department of Computer Science

# Define the Types - Types . hs



→ロト → 部ト → 重ト → 重 → りへで



Booleans	Adding Let	Conclusion	Booleans	Adding Let	Conclusion
<b>○</b> ●	000	0	00	000	0
0					

#### Eval - Bools, and, or

# Adding Comparisons



 Booleans
 Adding Let
 Conclusion
 Booleans
 Adding Let
 Conclusion

 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○
 ○

#### A Simple Let Expression

► We want to define local variables:

```
114> 3 + let x = 2 + 3 in x * x end
2 IntVal 28
```

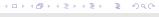
► Need two new Exp constructors.

# Coding Eval for Variables

► For variables, we look them up in the environment.

```
1 eval (VarExp var) env =
2   case lookup var env of
3     Just val -> val
4     Nothing -> IntVal 0
```





Booleans	Adding Let	Conclusion	Booleans	Adding Let	Conclusion
00	000	0	00	000	•
0			0		

# Coding Eval for Let

# 1 eval (LetExp var e1 e2) env = 2 let v1 = eval e1 env 3 in eval e2 (insert var v1 env)

▶ The insert var v1 env call acts like pushing a value onto a stack!

### **Next Time**

- ► You now have some interesting things for your interpreter.
- ► The reference implementation is in i4.
- ► We've also added a IfExp to the types if you want to try adding this.



