Clicker Questions
October 18
We want to call primitive, built-in procedures in the same way we call procedures represented as lambda expressions.

Consider \(( (\text{lambda} \ (x \ y) \ (+ \ x \ y)) \ (+ \ 2 \ 3) \ (* \ 3 \ 4))\). What does the lambda expression evaluate to?

A. 17
B. That C thingy. Contraption, Carbureator, something like that.
C. A closure with parameter list \((x \ y)\) and body \((+ \ x \ y)\)
D. A closure with parameter list \((x \ y)\), body \((+ \ x \ y)\), and the top-level environment.
Answer D: A closure with parameter list (x y), body (+ x y), and the top-level environment.
Consider \( C (+ 2 3) (* 3 4) \), where \( C \) is that closure we just discussed: parameter list \((x y)\), body \((+ x y)\), top-level environment. How do we evaluate \( C (+ 2 3) (* 3 4) \)?

A. Take the closure body \((+ x y)\), replace \(x\) by \((+ 2 3)\), replace \(y\) by \((* 3 4)\) and perform the computation.

B. Evaluate \((+ 2 3)\) to get 5, evaluate \((* 3 4)\) to get 12, replace \(x\) by 5 and \(y\) by 12 in \((+ x y)\) to get \((+ 5 12)\) and do the calculation.

C. Evaluate \((+ 2 3)\) to get 5, evaluate \((* 3 4)\) to get 12, extend the closure environment with bindings of \(x\) to 5 and \(y\) to 12, then evaluate \((+ x y)\) this extended environment.

D. Evaluate \((+ 2 3)\) to get 5, evaluate \((* 3 4)\) to get 12, extend the closure environment with bindings of \(x\) to 5 and \(y\) to 12, then evaluate \((+ x y)\) this extended environment.
Answer C: Evaluate \((+ 2 3)\) to get 5, evaluate \((* 3 4)\) to get 12, extend the **closure** environment with bindings of \(x\) to 5 and \(y\) to 12, then evaluate \((+ x y)\) this extended environment.
Now consider a primitive operation, like +. Do we need a closure for +?

A. Yes.
B. No.
C. Yes because we need closures for everything and somewhere there is a lambda expression for +/
D. No because we don't have any unbound variables in +.
Answer C: No because we don't have any unbound variables in +, or in any of the primitive functions.
How will we call a primitive function?

A. Evaluate the arguments, and pass their values to something that implements the primitive function.
B. Don't evaluate the arguments; pass their parse trees to something that implements the primitive function.
C. Turn the primitive function into a lambda expression and call it like we call other lambda expressions.
D. Write our parser so it immediately replaces an expression with a primitive function call, such as (+ 2 3), with its value 5.
Answer A: Evaluate the arguments, and pass their values to something that implements the primitive function.
'+' is a symbol; 'cons is a symbol; 'f and 'foo are symbols. How can we tell the difference between (+ 2 4), which calls a primitive function, and (f 2 4), which calls a procedure defined through a lambda exp?

A. Give the parser a list of all of the primitive functions. When it finds a symbol that represents a primitive function it can label the call as calling a primitive rather than a lambda expression.
B. Give the environment bindings for all of the primitive functions; when we look up a symbol in the environment, it will find the appropriate binding.
C. At the time we do the call, look to see if the function part is a closure or a symbol.
D. If we implement primitive functions as lambda expressions we don't need to tell the difference.
Answer B: Give the environment bindings for all of the primitive functions; when we look up a symbol in the environment, it will find the appropriate binding.