μScheme vs Impcore

New abstract syntax:
- LET (keyword, names, expressions, body)
- LAMBDA (formals, body)
- APPLY (exp, actuals)

New concrete syntax for LITERAL:

(quote S-expression)

' S-expression
Introduce local names into environment

(\texttt{let \ ([x1 \ e1]}
\quad \ldots \quad
\texttt{\ [xn \ en]})
\texttt{e})

\textbf{Square brackets mean the same as round, but are easier to see}
Function escapes!

```lisp
-> (define to-the-n-minus-k (n k)
    (let
      ([x-to-the-n-minus-k (lambda (x)
                          (- (exp x n) k))])
      x-to-the-n-minus-k))
-> (val x-cubed-minus-27 (to-the-n-minus-k 3 27))
-> (x-cubed-minus-27 2)
-19
```
No need to name the escaping function

```scheme
-> (define to-the-n-minus-k (n k)
    (lambda (x) (- (exp x n) k)))

-> (val x-cubed-minus-27 (to-the-n-minus-k 3 27))
-> (x-cubed-minus-27 2)
-19
```
The zero-finder

(define findzero-between (f lo hi)
  ; binary search
  (if (>= (+ lo 1) hi)
      hi
      (let ([mid (/ (+ lo hi) 2)])
        (if (< (f mid) 0)
            (findzero-between f mid hi)
            (findzero-between f lo mid))))
  (define findzero (f) (findzero-between f 0 100))
Cube root of 27 and square root of 16

-> (findzero (to-the-n-minus-k 3 27))
3

-> (findzero (to-the-n-minus-k 2 16))
4
Lambda questions

(define combine (p? q?)
  (lambda (x) (if (p? x) (q? x) #f)))

(define divvy (p? q?)
  (lambda (x) (if (p? x) #t (q? x))))

(val c-p-e (combine prime? even?))
(val d-p-o (divvy prime? odd?))

(c-p-e 9) == ?       (d-p-o 9) == ?
(c-p-e 8) == ?       (d-p-o 8) == ?
(c-p-e 7) == ?       (d-p-o 7) == ?
Lambda answers

(define combine (p? q?)
    (lambda (x) (if (p? x) (q? x) #f)))

(define divvy (p? q?)
    (lambda (x) (if (p? x) #t (q? x)))))

(val c-p-e (combine prime? even?))
(val d-p-o (divvy prime? odd?))

(c-p-e 9) == #f      (d-p-o 9) == #t
(c-p-e 8) == #f      (d-p-o 8) == #f
(c-p-e 7) == #f      (d-p-o 7) == #t