CSE 14A lecture 12!

- midterm recap
- environments
  (variables and local bindings in our interpreter)

  $3 + 4 \Rightarrow 7$

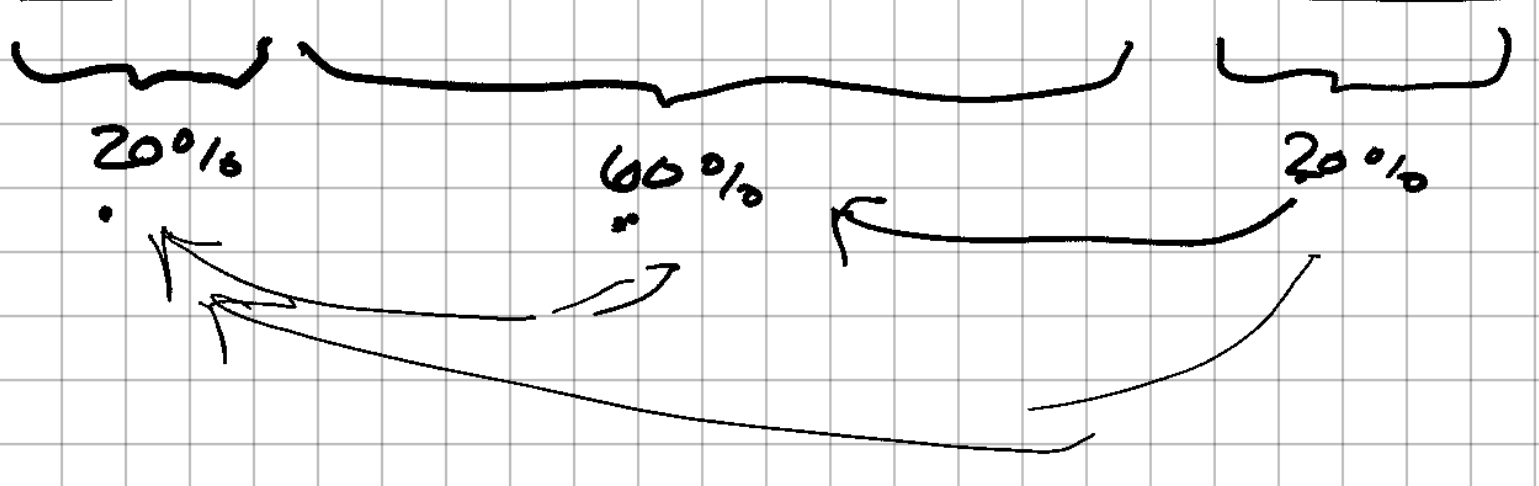
  $x + 4 \Rightarrow ?$

Midterm recap

20% — is kicking butt

20% - is really struggling

60% - so-so



1.

$(\lambda f\ g\ h \rightarrow f\ g)$
$\quad (\lambda x \rightarrow x)$
$\quad (\lambda y \rightarrow (\lambda z \rightarrow y))$
$\quad (\lambda q \rightarrow q)$

$\Rightarrow_b$

$(\lambda g\ h \rightarrow (\lambda x \rightarrow x)\ g)$
$\quad (\lambda y \rightarrow (\lambda z \rightarrow y))$
$\quad (\lambda q \rightarrow q)$

$\Rightarrow_b$

$(\lambda h \rightarrow \underline{(\lambda x \rightarrow x)\ (\lambda y \rightarrow (\lambda z \rightarrow y))})$
$\quad (\lambda q \rightarrow q)$

$\Rightarrow_b (\lambda x \rightarrow x)\ (\lambda y \rightarrow (\lambda z \rightarrow y))$

$\Rightarrow_b (\lambda y \rightarrow (\lambda z \rightarrow y))$

2.

How do you know when n is less than m? (or equal to)

SUB n m helps!

ISZ (SUB n m)

ITE (ISZ (SUB n m)) TRUE FALSE

3. 0, 1, 1, 2, 3, 5, . . .

Fib(n) = if n ≤ 1 then n
otherwise
Fib(n-1) + Fib(n-2)

\rec →
FIB1 = \n → ITE (LEQ n 1)
n
ADD (rec (DECR n)) (rec (SUB n 2))

FIB = Y FIB1

ITE (ISZ n)
ZERO
ITE (ISZ (DECR n))
ONE

LEQ n 2
ITE ~~~

4) How to think about these?
operationally or like a constraint solver

(a) [Bool]

True :: Bool

[False, True, False] :: [Bool]

(:) a → [a] → [a]

Bool → [Bool]

(:) True [False, True, False]

↑ Bool [Bool]

a → [a] → [a] ← [Bool]

(b) (a → b) → String

a → (b → String)

a → b → c → d

a → (b → (c → d))

(c) String

foldr :: (a → b → b) → b → [a] → b

(++) ""

(d) Bool → [Bool]

(e) Maybe String → String

(f) [a, b, c] ++ [d, e]
⇒ [a, b, c, d, e]

map :: (a → b) → [a] → [b]

\x y → x :: c → (d → c)

[Bool] (a → Bool)]

5)

a) [(False, "hi")]

b) \x → "charizard" (too general)

\x → if x
then "squirtle"
else "mew"

Bool→String

c) \f → [(f True) ++ "hi"]

d) a → a

(\x → x) (\x → x)

(a→b)

7)

```
Lam "x" (Lam "y"
          (App (Lam "z" (Var "y"))
               (App (Var "x") (Var "z"))))
```

8)

```
depth :: LCExpr → Int

depth (Var s)      = 1
depth (Lam id e)   = 1 + depth e
depth (App e1 e2)  = 1 + max (depth e1)
                             (depth e2)
```

9)

a)

```
(==) :: LCExpr → LCExpr → Bool
(==) e1 e2 = depth e1 == depth e2
```

b)

```
(<=) e1 e2 = depth e2 <= depth e2
```

10)

```
freeVars :: LCExpr → [String]
freeVars (Var s)   = [s]
freeVars (Lam s e) : freeVars e \\ [s]
freeVars (App e1 e2) : nub (freeVars e1 ++
                            freeVars e2)
```