1-1

compiler : A → B

A is a program in lang X
B is a program in lang Y

gcc : C-prog → X-86-prog

interp : A → answer

When is a compiler right?

$$\begin{tikzcd}
A \arrow[r, "\text{interp}_A"] \arrow[d, "\text{compile}_{A\to B}"] & \text{Ans}_A \arrow[d, bend left, "\text{convert}_{A\to B}"] \\
B \arrow[r, "\text{interp}_B"] & \text{Ans}_B \arrow[u, "\text{convert}_{B\to A}"]
\end{tikzcd}$$

diagram
"commutes"

Log

"$\forall a \in A.\ \text{interp}_A(a) = \overset{\text{convert}_{A\to B}}{\text{interp}_B}(\text{compile}_{A\to B}(a))$"

When is a compiler good?

some other metric (time, space) m (measures goodness)
m(A) < m(compile(A))

large gap → good compiler

"optimizations" ways of improving m

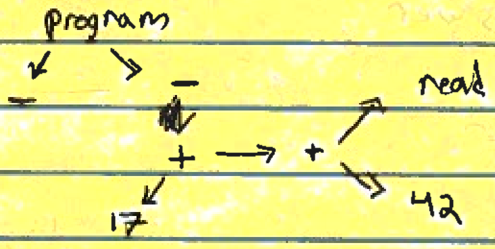
1-2/ R_0 : P = program info e

e = number | (- e) | (+ e e) | (read)

" return - 17 + read() + 42; "

→

* (program _ (- (+ 17 (+ (read) 42))))



```
B* A(B* l, B* r) { return new Add(l, r); };

new Program(NULL, new Neg(new Add(new Num(17),
                                  new Add(new Read(),
                                          new Num(42)));
```

Program : info-ptr × Expr → Prog
Neg : Expr → Expr
Add : Expr × Expr → Expre
Read : → Expr
Num : int → Expr

interp (program _ e) = interp (e)   interp : R_0 → int
interp (Num n) = n
interp (Neg e) = -1 * interp (e)
interp (Add e1 e2) = interp (e1) + interp (e2) ←
interp (Read) = ask the user for a number

(+ (read) (- (read)))

```
class Expr {
  virtual int interp() = 0; }
```

```
class Add {
  Expr *lhs, *rhs;
  int interp() {
     return lhs->interp() + rhs->interp(); }
```

3)

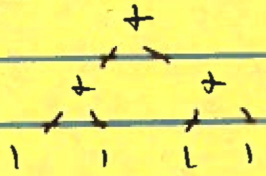
2^n : int → R_0

interp (2^n m) = 2^m

2^n 0 = 1

2^n (1+m) = (+ (2^n m) (2^n m))

2^n 2 =



randp : int → R_0

randp 0 = flip a coin head → (read)
tail → number ∈ [0, 1024)

randp (1+n) = flip a coin head → − (randp n)
tail → + (randp n)
(randp n)

opt : R_0 → R_0

opt (+ 2 3) = 5

opt (− (− 7)) = 7

(− (− e)) = opt e

opt (+ 7 (+ read 8)) = (+ 15 read)

opt n = n

opt read = read

opt (+ e_1 e_2) = e'_1 = opt e_1 , e'_2 = opt e_2

cases e'_1 , e'_2 ~~40% done~~

n, m → n $\hat{+}$ m

n, (+ m e) → (+ (n+m) e)

opt (− e_a) = e'_a = opt e_a 80%

cases e'_a n → −n | (− e) → e

(+ [illegible] −en) → (+ −n (− en))