# Leveraging Live Programming in the Classroom

## an experience report

Alex Warth

Y Research

UCLA

# Sfruttando la Programmazione Live in aula

## un rapporto di esperienza

Alessandro Warth

Y Research

UCLA


I'M WORKING
ON A COMPILER!


I'M WRITING A SEARCH ENGINE!


I WORK AT
GOOGLE

# Problems

- Our tools and languages are limited
- Real-world programmers won't use (or don't believe in) better tools and languages


THAT'S REALLY COOL, BUT I'LL PROBABLY STICK WITH JAVA.

First education,
then...

THE WORLD!

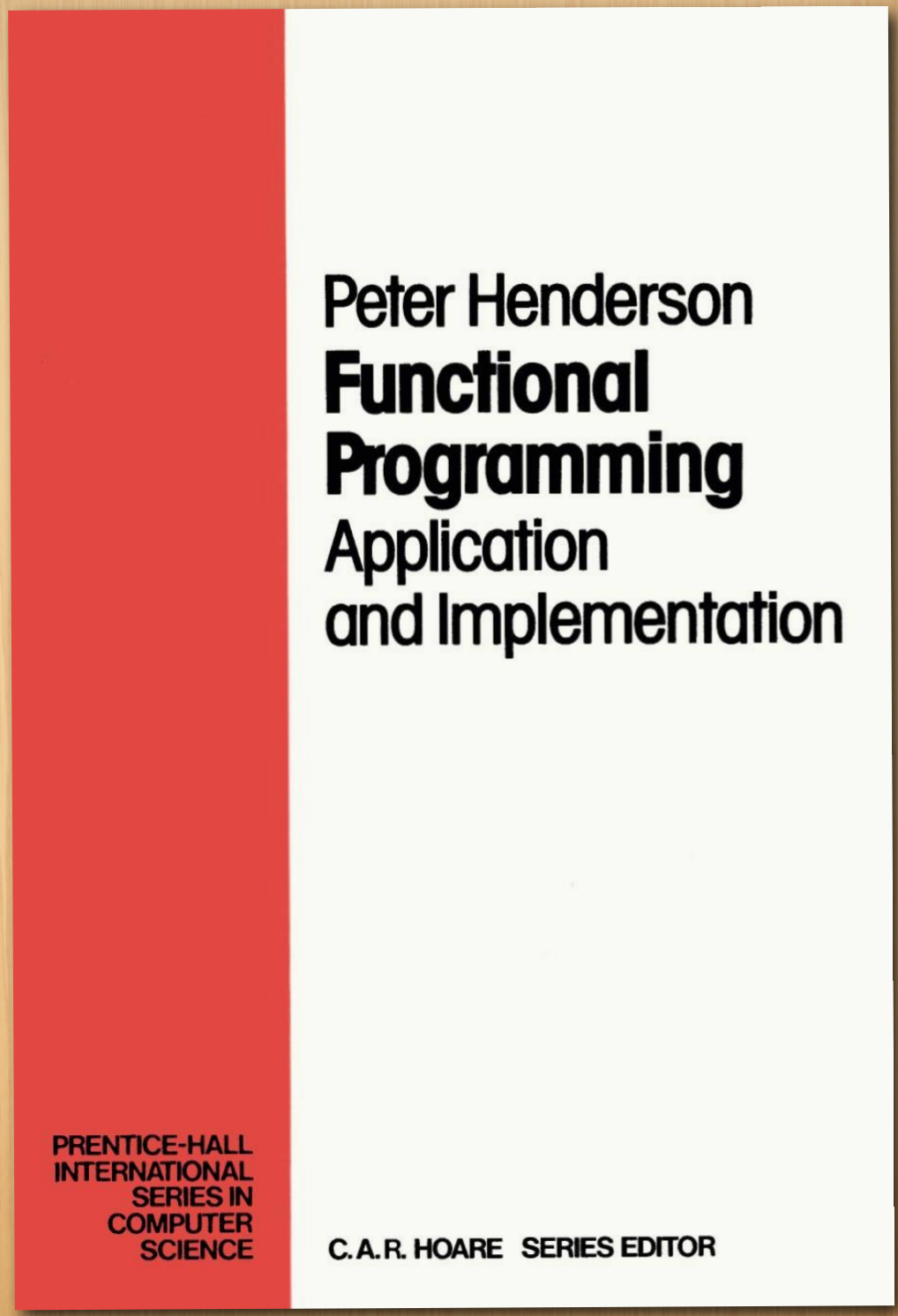
First education, then...

THE WORLD!

(muhahaha)

# Problems

- Our tools and languages are limited
- Real-world programmers won't use (or don't believe in) better tools and languages


Peter Henderson
Functional Programming
Application and Implementation
PRENTICE-HALL INTERNATIONAL SERIES IN COMPUTER SCIENCE
C. A. R. HOARE SERIES EDITOR

```
(LETREC COMPILE
  (COMPILE LAMBDA (E)
    (COMP E (QUOTE NIL) (QUOTE (4 21))))
  (COMP LAMBDA (E N C)
    (IF (ATOM E)
        (CONS (QUOTE 1) (CONS (LOCATION E N) C))
    (IF (EQ (CAR E) (QUOTE QUOTE))
        (CONS (QUOTE 2) (CONS (CAR (CDR E)) C))
    (IF (EQ (CAR E) (QUOTE ADD))
        (COMP (CAR (CDR E)) N (COMP (CAR (CDR (CDR E))) N (CONS (QUOTE 15) C)))
    (IF (EQ (CAR E) (QUOTE SUB))
        (COMP (CAR (CDR E)) N (COMP (CAR (CDR (CDR E))) N (CONS (QUOTE 16) C)))
    (IF (EQ (CAR E) (QUOTE MUL))
        (COMP (CAR (CDR E)) N (COMP (CAR (CDR (CDR E))) N (CONS (QUOTE 17) C)))
    (IF (EQ (CAR E) (QUOTE DIV))
        (COMP (CAR (CDR E)) N (COMP (CAR (CDR (CDR E))) N (CONS (QUOTE 18) C)))
    (IF (EQ (CAR E) (QUOTE REM))
        (COMP (CAR (CDR E)) N (COMP (CAR (CDR (CDR E))) N (CONS (QUOTE 19) C)))
    (IF (EQ (CAR E) (QUOTE LEQ))
        (COMP (CAR (CDR E)) N (COMP (CAR (CDR (CDR E))) N (CONS (QUOTE 20) C)))
    (IF (EQ (CAR E) (QUOTE EQ))
        (COMP (CAR (CDR E)) N (COMP (CAR (CDR (CDR E))) N (CONS (QUOTE 14) C)))
    (IF (EQ (CAR E) (QUOTE CAR))
        (COMP (CAR (CDR E)) N (CONS (QUOTE 10) C))
    (IF (EQ (CAR E) (QUOTE CDR))
        (COMP (CAR (CDR E)) N (CONS (QUOTE 11) C))
    (IF (EQ (CAR E) (QUOTE ATOM))
        (COMP (CAR (CDR E)) N (CONS (QUOTE 12) C))
    (IF (EQ (CAR E) (QUOTE CONS))
        (COMP (CAR (CDR (CDR E))) N (COMP (CAR (CDR E)) N (CONS (QUOTE 13) C)))
    (IF (EQ (CAR E) (QUOTE IF))
        (LET (COMP (CAR (CDR E)) N (CONS (QUOTE 8)
                                         (CONS THENPT (CONS ELSEPT C))))
          (THENPT COMP (CAR (CDR (CDR E))) N (QUOTE (9)))
          (ELSEPT COMP (CAR (CDR (CDR (CDR E)))) N (QUOTE  (9)))  )
    (IF (EQ (CAR E) (QUOTE LAMBDA))
        (LET (CONS (QUOTE 3) (CONS BODY C))
          (BODY COMP (CAR (CDR (CDR E))) (CONS (CAR (CDR E)) N)
                                         (QUOTE (5))) )
    (IF (EQ (CAR E) (QUOTE LET))
         (LET  (LET  (COMPLIS ARGS N (CONS (QUOTE 3)
                                           (CONS BODY (CONS (QUOTE 4) C))))
                     (BODY COMP (CAR (CDR E)) M (QUOTE (5))))
           (M CONS (VARS (CDR (CDR E))) N)
           (ARGS EXPRS (CDR (CDR E))))
    (IF (EQ (CAR E) (QUOTE LETREC))
         (LET  (LET  (CONS (QUOTE 6) (COMPLIS ARGS M
                                           (CONS (QUOTE 3) (CONS BODY (CONS (QUOTE 7) C)))))
                     (BODY COMP (CAR (CDR E)) M (QUOTE (5))))
           (M CONS (VARS (CDR (CDR E))) N)
           (ARGS EXPRS (CDR (CDR E))))
    (COMPLIS (CDR E) N (COMP (CAR E) N (CONS (QUOTE 4) C))))))))))))))))))))))))))))
  (COMPLIS LAMBDA (E N C)
    (IF (EQ E (QUOTE NIL)) (CONS (QUOTE 2) (CONS (QUOTE NIL) C))
        (COMPLIS (CDR E) N (COMP (CAR E) N (CONS (QUOTE 13) C)))))
  (LOCATION LAMBDA (E N)
    (LETREC
      (IF (MEMBER E (CAR N)) (CONS (QUOTE 0) (POSN E (CAR N)))
          (INCAR (LOCATION E (CDR N))))
    (MEMBER LAMBDA (E N)
          (IF (EQ N (QUOTE NIL)) (QUOTE F)
          (IF (EQ E (CAR N)) (QUOTE T) (MEMBER E (CDR N)))))
    (POSN LAMBDA (E N)
      (IF (EQ E (CAR N)) (QUOTE 0) (ADD (QUOTE 1) (POSN E (CDR N)))))
    (INCAR LAMBDA (L) (CONS (ADD (QUOTE 1) (CAR L)) (CDR L)))))
  (VARS LAMBDA (D)
    (IF (EQ D (QUOTE NIL)) (QUOTE NIL)
        (CONS (CAR (CAR D)) (VARS (CDR D)))))
  (EXPRS LAMBDA (D)
    (IF (EQ D (QUOTE NIL)) (QUOTE NIL)
        (CONS (CDR (CAR D)) (EXPRS (CDR D))))))
```



```
(6 2 NIL 3 (1 (0 . 0) 2 NIL 14 8 (2 NIL 9) (2 NIL 1 (0 . 0) 11 13 1 (1 . 5) 4 1
(0 . 0) 10 11 13 9) 5) 13 3 (1 (0 . 0) 2 NIL 14 8 (2 NIL 9) (2 NIL 1 (0 . 0) 11
13 1 (1 . 4) 4 1 (0 . 0) 10 10 13 9) 5) 13 3 (6 2 NIL 3 (1 (0 . 0) 11 2 1 1 (0
. 0) 10 15 13 5) 13 3 (1 (0 . 0) 1 (0 . 1) 10 14 8 (2 0 9) (2 1 2 NIL 1 (0 . 1)
11 13 1 (0 . 0) 13 1 (1 . 1) 4 15 9) 5) 13 3 (1 (0 . 1) 2 NIL 14 8 (2 F 9) (1
(0 . 0) 1 (0 . 1) 10 14 8 (2 T 9) (2 NIL 1 (0 . 1) 11 13 1 (0 . 0) 13 1 (1 . 0)
4 9) 9) 5) 13 3 (2 NIL 1 (1 . 1) 10 13 1 (1 . 0) 13 1 (0 . 0) 4 8 (2 NIL 1 (1 .
1) 10 13 1 (1 . 0) 13 1 (0 . 1) 4 2 0 13 9) (2 NIL 2 NIL 1 (1 . 1) 11 13 1 (1 .
0) 13 1 (2 . 3) 4 13 1 (0 . 2) 4 9) 5) 7 5) 13 3 (1 (0 . 0) 2 NIL 14 8 (1 (0 .
2) 2 NIL 13 2 2 13 9) (2 NIL 2 NIL 1 (0 . 2) 2 13 13 13 1 (0 . 1) 13 1 (0 . 0)
10 13 1 (1 . 1) 4 13 1 (0 . 1) 13 1 (0 . 0) 11 13 1 (1 . 2) 4 9) 5) 13 3 (1 (0
. 0) 12 8 (1 (0 . 2) 2 NIL 1 (0 . 1) 13 1 (0 . 0) 13 1 (1 . 3) 4 13 2 1 13 9)
(1 (0 . 0) 10 2 QUOTE 14 8 (1 (0 . 2) 1 (0 . 0) 11 10 13 2 2 13 9) (1 (0 . 0)
10 2 ADD 14 8 (2 NIL 2 NIL 1 (0 . 2) 2 15 13 13 1 (0 . 1) 13 1 (0 . 0) 11 11 10
13 1 (1 . 1) 4 13 1 (0 . 1) 13 1 (0 . 0) 11 10 13 1 (1 . 1) 4 9) (1 (0 . 0) 10
2 SUB 14 8 (2 NIL 2 NIL 1 (0 . 2) 2 16 13 13 1 (0 . 1) 13 1 (0 . 0) 11 11 10 13
1 (1 . 1) 4 13 1 (0 . 1) 13 1 (0 . 0) 11 10 13 1 (1 . 1) 4 9) (1 (0 . 0) 10 2
MUL 14 8 (2 NIL 2 NIL 1 (0 . 2) 2 17 13 13 1 (0 . 1) 13 1 (0 . 0) 11 11 10 13 1
(1 . 1) 4 13 1 (0 . 1) 13 1 (0 . 0) 11 10 13 1 (1 . 1) 4 9) (1 (0 . 0) 10 2 DIV
14 8 (2 NIL 2 NIL 1 (0 . 2) 2 18 13 13 1 (0 . 1) 13 1 (0 . 0) 11 11 10 13 1 (1
. 1) 4 13 1 (0 . 1) 13 1 (0 . 0) 11 10 13 1 (1 . 1) 4 9) (1 (0 . 0) 10 2 REM 14
8 (2 NIL 2 NIL 1 (0 . 2) 2 19 13 13 1 (0 . 1) 13 1 (0 . 0) 11 11 10 13 1 (1 .
1) 4 13 1 (0 . 1) 13 1 (0 . 0) 11 10 13 1 (1 . 1) 4 9) (1 (0 . 0) 10 2 LEQ 14 8
(2 NIL 2 NIL 1 (0 . 2) 2 20 13 13 1 (0 . 1) 13 1 (0 . 0) 11 11 10 13 1 (1 . 1)
4 13 1 (0 . 1) 13 1 (0 . 0) 11 10 13 1 (1 . 1) 4 9) (1 (0 . 0) 10 2 EQ 14 8 (2
NIL 2 NIL 1 (0 . 2) 2 14 13 13 1 (0 . 1) 13 1 (0 . 0) 11 11 10 13 1 (1 . 1) 4
13 1 (0 . 1) 13 1 (0 . 0) 11 10 13 1 (1 . 1) 4 9) (1 (0 . 0) 10 2 CAR 14 8 (2
NIL 1 (0 . 2) 2 10 13 13 1 (0 . 1) 13 1 (0 . 0) 11 10 13 1 (1 . 1) 4 9) (1 (0 .
0) 10 2 CDR 14 8 (2 NIL 1 (0 . 2) 2 11 13 13 1 (0 . 1) 13 1 (0 . 0) 11 10 13 1
(1 . 1) 4 9) (1 (0 . 0) 10 2 ATOM 14 8 (2 NIL 1 (0 . 2) 2 12 13 13 1 (0 . 1) 13
1 (0 . 0) 11 10 13 1 (1 . 1) 4 9) (1 (0 . 0) 10 2 CONS 14 8 (2 NIL 2 NIL 1 (0 .
2) 2 13 13 13 1 (0 . 1) 13 1 (0 . 0) 11 10 13 1 (1 . 1) 4 13 1 (0 . 1) 13 1 (0
. 0) 11 11 10 13 1 (1 . 1) 4 9) (1 (0 . 0) 10 2 IF 14 8 (2 NIL 2 NIL 2 (9) 13 1
(0 . 1) 13 1 (0 . 0) 11 11 11 10 13 1 (1 . 1) 4 13 2 NIL 2 (9) 13 1 (0 . 1) 13
1 (0 . 0) 11 11 10 13 1 (1 . 1) 4 13 3 (2 NIL 1 (1 . 2) 1 (0 . 1) 13 1 (0 . 0)
13 2 8 13 13 1 (1 . 1) 13 1 (1 . 0) 11 10 13 1 (2 . 1) 4 5) 4 9) (1 (0 . 0) 10
2 LAMBDA 14 8 (2 NIL 2 NIL 2 (5) 13 1 (0 . 1) 1 (0 . 0) 11 10 13 13 1 (0 . 0)
11 11 10 13 1 (1 . 1) 4 13 3 (1 (1 . 2) 1 (0 . 0) 13 2 3 13 5) 4 9) (1 (0 . 0)
10 2 LET 14 8 (2 NIL 2 NIL 1 (0 . 0) 11 11 13 1 (1 . 5) 4 13 1 (0 . 1) 2 NIL 1
(0 . 0) 11 11 13 1 (1 . 4) 4 13 13 3 (2 NIL 2 NIL 2 (5) 13 1 (0 . 0) 13 1 (1 .
0) 11 10 13 1 (2 . 1) 4 13 3 (2 NIL 1 (2 . 2) 2 4 13 1 (0 . 0) 13 2 3 13 13 1
(2 . 1) 13 1 (1 . 1) 13 1 (3 . 2) 4 5) 4 5) 4 9) (1 (0 . 0) 10 2 LETREC 14 8 (2
NIL 2 NIL 1 (0 . 0) 11 11 13 1 (1 . 5) 4 13 1 (0 . 1) 2 NIL 1 (0 . 0) 11 11 13
1 (1 . 4) 4 13 13 3 (2 NIL 2 NIL 2 (5) 13 1 (0 . 0) 13 1 (1 . 0) 11 10 13 1 (2
. 1) 4 13 3 (2 NIL 1 (2 . 2) 2 7 13 1 (0 . 0) 13 2 3 13 13 1 (1 . 0) 13 1 (1 .
1) 13 1 (3 . 2) 4 2 6 13 5) 4 5) 4 9) (2 NIL 2 NIL 1 (0 . 2) 2 4 13 13 1 (0 .
1) 13 1 (0 . 0) 10 13 1 (1 . 1) 4 13 1 (0 . 1) 13 1 (0 . 0) 11 13 1 (1 . 2) 4
9) 9) 9) 9) 9) 9) 9) 9) 9) 9) 9) 9) 9) 9) 9) 9) 9) 9) 9) 5) 13 3 (2 NIL 2 (4 21) 13 2
NIL 13 1 (0 . 0) 13 1 (1 . 1) 4 5) 13 3 (1 (0 . 0) 5) 7 4 21)
```

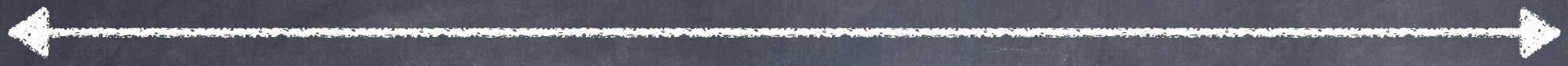
CS239, Lec 1, Spring 2008
MW 12-1:50pm
Boelter 9436

Title: Programming Language Design Laboratory

Instructors: Todd Millstein and Alan Kay

Description:

This seminar will explore the principles and practice of programming language design, with a goal toward investigating better (more expressive, easier to learn, easier to extend, more reliable) ways to program than current approaches. The course will include readings of technical papers and discussions about interesting and unusual language designs, both historical and from the current research literature. For the main work of the course, students will work in groups to design, implement, and possibly formalize their own language or software development tool.



sketch
(generate lots of ideas)

prototype
(validate ideas)

implementation

sketch
(generate lots of ideas)

prototype
(validate ideas)

implementation

Compilers

sketch
(generate lots of ideas)

prototype
(validate ideas)

implementation

Compilers

PL Design Seminar

sketch
(generate lots of ideas)

prototype
(validate ideas)

implementation

Compilers

CS137A/237A
Prototyping PLs

PL Design Seminar

# "Prototyping PLs"

Lightweight techniques for implementing PLs

# "Prototyping PLs"

Lightweight techniques for implementing PLs

- Traditional PL implementation:
  optimize for efficiency

# "Prototyping PLs"

Lightweight techniques for implementing PLs

- Traditional PL implementation: optimize for efficiency

# "Prototyping PLs"

Lightweight techniques for implementing PLs

- Traditional PL implementation: optimize for efficiency
- In the new class: optimize for simplicity

# Why Optimize for Simplicity?

- get something working quickly
- easier to modify / extend it / ...
  (try out different design decisions)
- can always optimize later,
  if and when necessary

# Why Optimize for Simplicity?

- get something working quickly
- easier to modify / extend it / ...
  (try out different design decisions)
- can always optimize later,
  if and when necessary

(good advice in general)

# Approach

- Students get experience with several prototyping styles & techniques:
  - interpreters
  - source-to-source translators
  - embedded / internal DSLs
- By "prototyping" some PLs that they already know: functional, OO, logic, ...

# Approach

- Students get experience with several prototyping styles & techniques:
  - interpreters
  - source-to-source translators
  - embedded / internal DSLs
- By "prototyping" some PLs that they already know: functional, OO, logic, ...

("Multi-paradigm Henderson")

# Undergrad PLs Course "Under the Hood"

Students get a deeper understanding of things like:

- Closures
- Classes and dynamic dispatch
- Prolog-style search and unification
- ...

# Our Framework

# Principles

- Provide instant gratification whenever possible
- Avoid unnecessary friction (no "command-line bullshittery")

# calculator language

(interpreter)

# functional language

(interpreter)

# OO language

(source-to-source translator)

# Principles

- Provide instant gratification whenever possible
- Avoid unnecessary friction (no "command-line bullshittery")

# The Framework in Lectures

(worlds demo)

# Results

# Excerpts from Course Evaluations

# Excerpts from Course Evaluations

BEST CLASS EVER!

# Excerpts from Course Evaluations

AWESOME COURSE!

# Excerpts from Course Evaluations

Really awesome course!

# Excerpts from Course Evaluations

EXCELLENT COURSE, AMONG THE BEST I'VE EVER TAKEN.

# Excerpts from Course Evaluations

Eye-opening and transformative.

# Excerpts from Course Evaluations

I FINALLY UNDERSTAND HOW CLOSURES WORK!

# Excerpts from Course Evaluations

I think the tools provided were extremely helpful. [...] They got me to not be afraid of hard topics and approach them in a casual and fun way, instead of being burdened by consoles, compilers, and debuggers.

# Excerpts from Course Evaluations

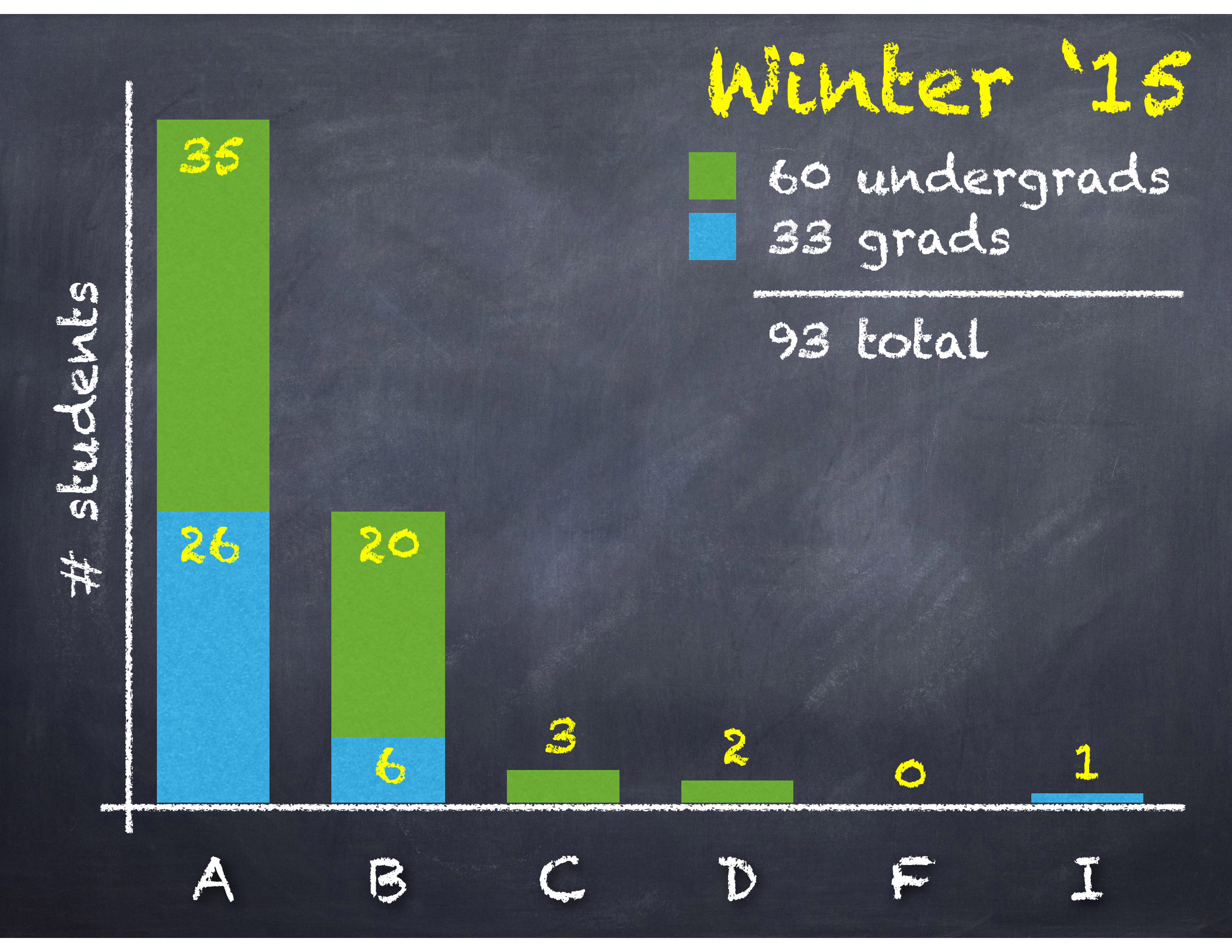
THE INTERFACE THAT PROFESSOR WARTH MADE FOR THE HOMEWORK WAS GENIUS AND AMAZING.

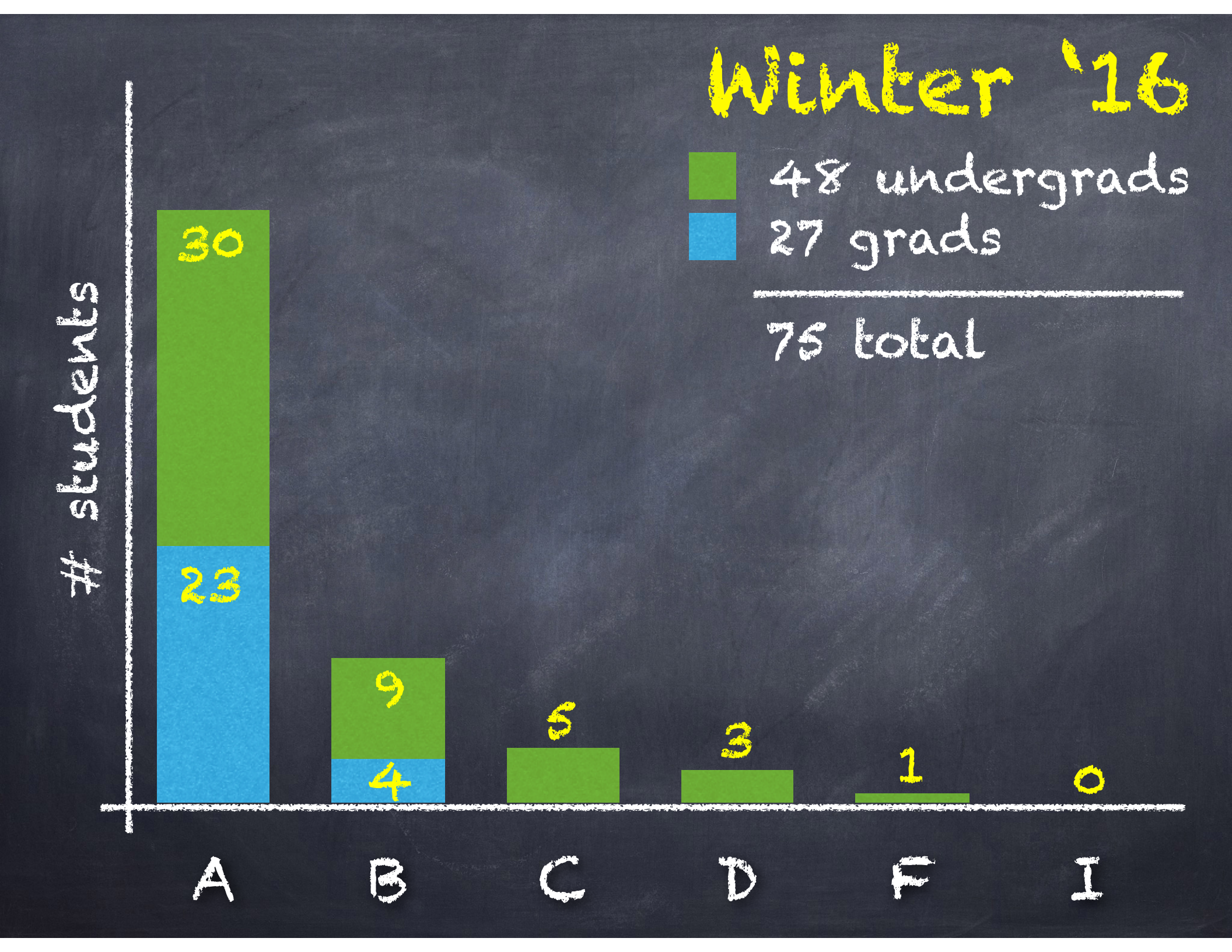
# Excerpts from

The assignments were great.
And I really mean top notch.
Well structured, [...], good coverage of different language paradigms and implementation strategies, and rich enough features but without anything to distract from the main point of the project.

# Excerpts from

Homeworks / projects were some of the most interesting and fun projects that I have done in my undergraduate career.

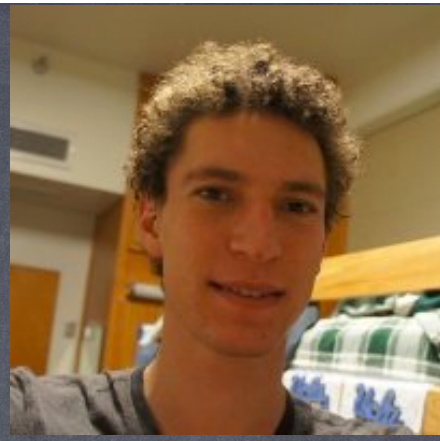

Winter '15
60 undergrads
33 grads
93 total
# students
35
26
20
6
3
2
0
1
A
B
C
D
F
I


Winter '16
48 undergrads
27 grads
75 total
# students
30
23
9
4
5
3
1
0
A
B
C
D
F
I

# Extra-Credit Projects

# Jacob Sharf

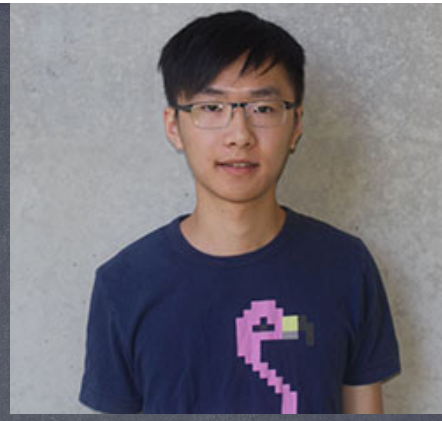
(3rd year undergrad)



```
introduce x, y
such that x + y * y = 7
in x / y
```

# Andrew Lai

(3rd year undergrad)



# Prolog Visualizer

# Results
(Discussion)

I Googled "Who Gives a Shit?"

My name wasn't in the search results.

# Next Steps...

**concrete syntax**

```
let f = fun x -> x * x in
  f 5 + 17
```

**abstract syntax**

```
new Let(
  new Var("f"),
  new Fun(
    new Var("x"),
    new BinOp(
      "*",
      new Var("x"),
      new Var("x"))),
  new BinOp(
    "+"
```

**result**

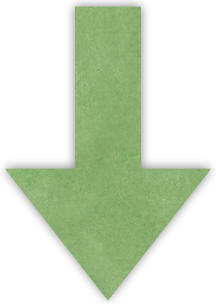
```
42
```

**concrete syntax**

```
let f = fun x -> x * x in
  f 5 + 17
```

**abstract syntax**

```
new Let(
  new Var("f"),
  new Fun(
    new Var("x"),
    new BinOp(
      "*",
      new Var("x"),
      new Var("x"))),
  new BinOp(
    "+"
```



**result**

```
42
```

# What about other classes?

- Algorithms?
- Intro to programming?
  - ... for college students
  - ... for high-school students
  - ... for middle-school students?

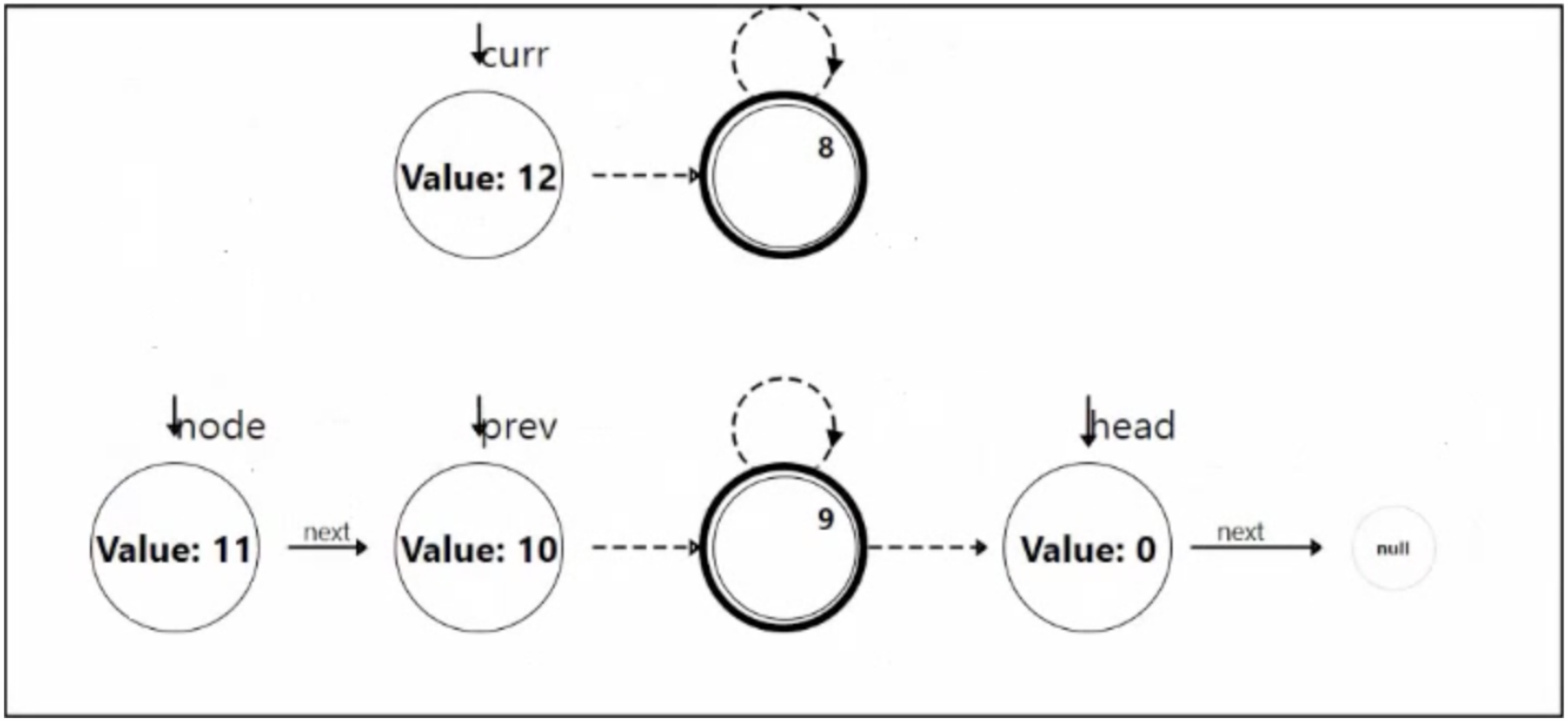
```
function binarySearch(key, array) {

  var low = 0;
  var high = array.length - 1;

  while (low <= high) {

    var mid = floor((low + high) / 2);
    var value = array[mid];

    if (value < key) {
      low = mid + 1;
    } else if (value > key) {
      high = mid - 1;
    } else {
      return mid;
    }
  }

  return -1;
}
```

```
   key = 'g'
array = ['a','b','c','d','e','f']
   low = 0
  high = 5

   low =  0  |  3  |  5
  high =  5  |  5  |  5
   mid =  2  |  4  |  5
value = 'c' | 'e' | 'f'


   low =  3  |  5  |  6

  return -1
```

[Bret Victor, CUSEC ’12]

```
3  public class Reverse {
4      public static
5      Node reverse(Node head) {
6          Node node = null;
7          Node prev = null;
8          Node curr = head;
9          while(curr.next != null){
10             node = curr;
11             curr = curr.next;
12             node.next = prev;
13             prev = node;
14         }
15         return head;
16     }
17 }
18
```

[Ou et al., CHI ’15]



# A Call to Action

The Live Programming / PX community has a chance to make a huge difference in the classroom!

- Consider applications of our research to educational programming environments
- Better yet, let's build educational environments and languages

# One last thing:

Interested in teaching this class at your university?

Come talk to me!