CSCI 334:
Principles of Programming Languages

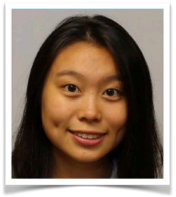
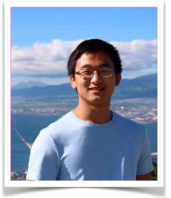
Lecture 3: PL Fundamentals

Instructor: Dan Barowy

Williams

Announcements

Colloquium today:
thesis presentations in TCL 202, 2:30pm

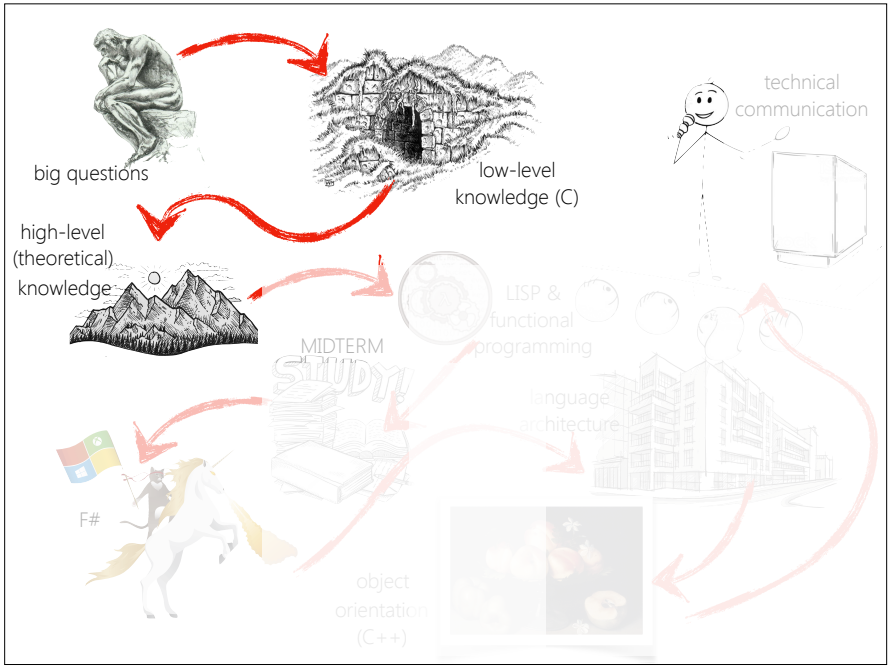

(they've all worked really hard—
show support for your fellow CS Ephs!)

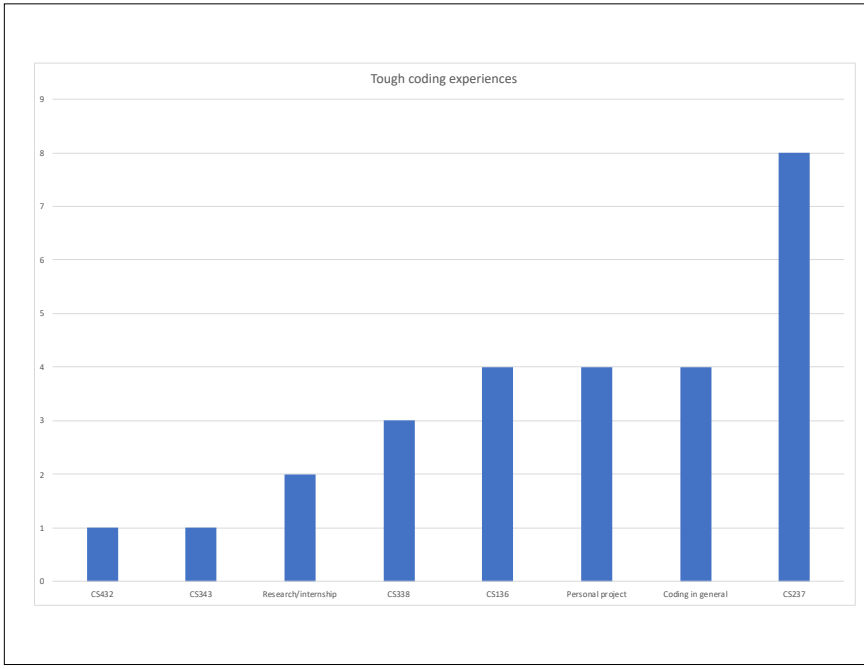
Announcements

- Next quiz is on Tuesday
- Quizzes cover previous week's material
- Therefore, the quiz will be on…
- … the lambda calculus.

Outline

- Short discussion of Pirsig
- Stack drawing
- Lab 2 hint
- PL foundations





Activity

```
#include <stdio.h>

void add(int *x, int *y, int *z) {
  *z = *x + *y;
}

int main() {
  int x = 1;
  int y = 3;
  int z;
  add(&x, &y, &z);
  return z;
}
```

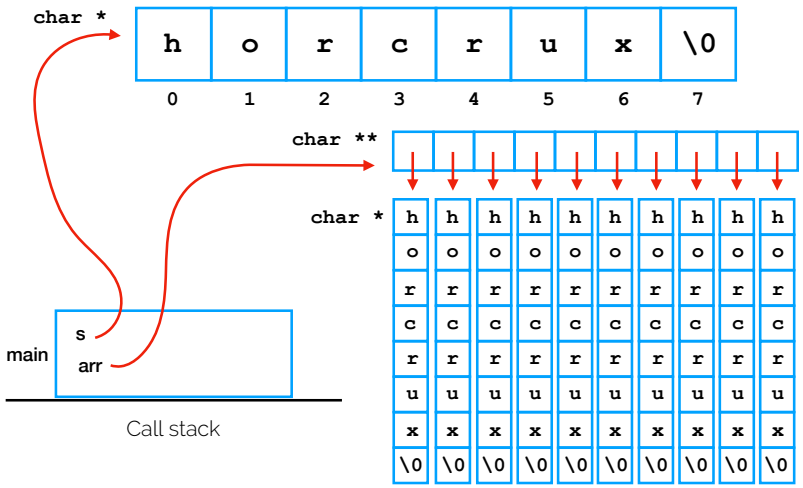
2 → `}`

1 → `int z;`

3 → `return z;`

Diagram the stack and variables when the program is at the three points.

Generate a random C string (a `char *`) and copy it to every element of an array of length 10. Make sure that every element is a *copy of the string, not a copy of the pointer*. The array should have *manual* storage duration. Print the array. Be sure to deallocate when done.



Activity

Write a function `swap` that swaps the values of `x` and `y`.

Start by drawing a picture of what you want.

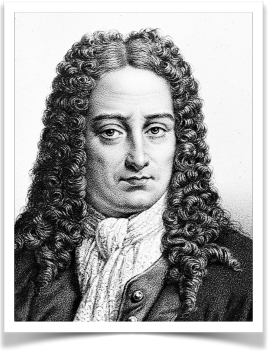
```
#include <stdio.h>

void swap(int *a, int *b) {
  ???
}

int main() {
  int x = 1;
  int y = 2;
  swap(&x, &y);
  printf("x = %d, y = %d\n", x, y);
  return 0;
}
```

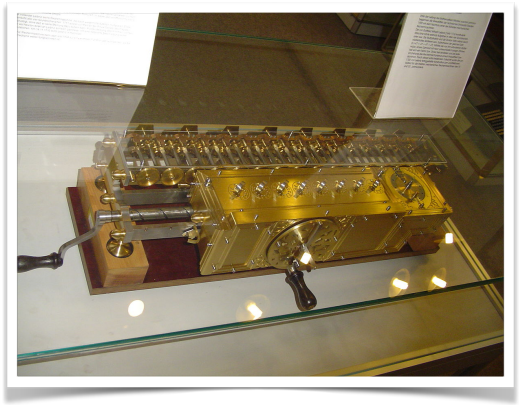
The Dream

"I thought again about my early plan of a new language or writing-system of reason, which could serve as a communication tool for all different nations... If we had such an universal tool, we could discuss the problems of the metaphysical or the questions of ethics in the same way as the problems and questions of mathematics or geometry. That was my aim: Every misunderstanding should be nothing more than a miscalculation (...), easily corrected by the grammatical laws of that new language. Thus, in the case of a controversial discussion, two philosophers could sit down at a table and just calculating, like two mathematicians, they could say, 'Let us check it up ...'"

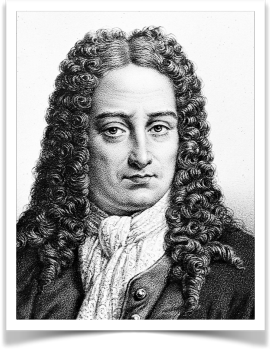


Wilhelm Gottfried Leibniz

The Dream



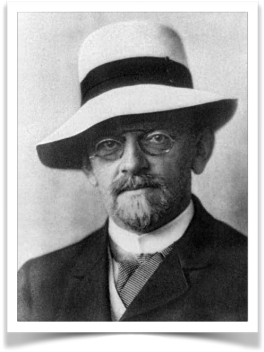
"stepped reckoner"



Wilhelm Gottfried Leibniz

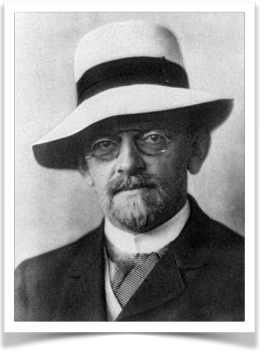
"What is the answer to the ultimate question of life, the universe, and everything?

What is computable?



- Hilbert: Is there an algorithm that can decide whether a logical statement is valid?
- "Entscheidungsproblem" (literally "decision problem")
- Leibniz thought so!

What is computable?



- Why do we care?
- f(x) = x + 1
- We can clearly do this with pencil and paper.
- $\int 6x\,dx$
- Also computable, in a different manner.
- We care because the computable functions can be done on a computer.

Lambda calculus

- Invented by Alonzo Church in order to solve the Entscheidungsproblem.
- Short answer to Hilbert's question: no.
- Proof: No algorithm can decide equivalence of two arbitrary **λ**-calculus expressions.
- By implication: no algorithm can determine whether an arbitrary logical statement is valid.

Lambda calculus grammar

```
<expr>  ::= <var>
          |  <abs>
          |  <app>
<var>   ::= x
<abs>   ::= λ<var>.<expr>
<app>   ::= <expr><expr>
```

What is a variable?

```
<var>    ::= x
```

It's just a value.

What is an abstraction?

```
<abs>    ::= λ<var>.<expr>
```

It's a function definition

```
def foo(x):
  <expr>
```

What is an application?

```
<app>    ::= <expr><expr>
```

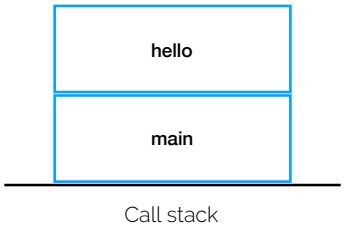
It's a "function call"

```
foo(2)
```

```
<expr><expr>
```

function argument

Evaluation: You know how C does it




```
#include <stdio.h>

void hello() {
  printf("Hello world!\n");
}

int main() {
  hello();
  return 0;
}
```

Evaluation: Lambda calculus is like algebra

`(λx.x)x`

Evaluation consists of simplifying an expression using text substitution.

Only two simplification rules:

α-reduction

β-reduction

Recap & Next Class

Today we covered:

More boxes and arrows

PL Foundations

Next class:

Foundations

Project idea:

`http://worrydream.com/AlligatorEggs/`