Node find (Node prev, Node cur, int key) {
    while (cur.key < key) {
        prev = cur;
        cur = cur.next;
    }
    return cur;
}
A revolution in the making
A revolution in the making

Program analysis technology is transformative
- From automatic bug finders and other software quality tools
- To advanced language features to ensure safety and modularity
- To formal verification techniques that guarantee the absence of bugs in critical software
- To sophisticated optimizations that make high-level languages practical
- Even synthesis techniques that automate challenging programming tasks

Learn the fundamental principles behind this technology!
- And take care of one of your TQE requirements!
What this course is about

The central theme of this course:

How do we get tools to reason about a program’s behavior

Before we get to that though...

- we must be able to reason about program behaviors ourselves
What this course is about

Four fundamental techniques to reason about programs
- Type theory
- Axiomatic Semantics
- Abstract Interpretation
- Model Checking

Along the way you will also learn about
- Lambda calculus
- Haskell
- Ocaml
- Automated reasoning tools like the SMT solver Yices and the interactive theorem prover Coq

You will also implement your own program analysis
- There will be a fair amount of homework
- But it will be worth it!
6 Homework Assignments

Pset 1
- Practice functional programming
- Build some Lambda Calculus Interpreters

Pset 2
- Practice more functional programming
- Practice writing proofs in Coq
- Implement a type inference engine

Pset 3
- How to make formal arguments about the properties of a type system
- Prove interesting properties in Coq

Pset 4
- Learn about SMT solvers
- Implement your own verifier for simple C programs
Homework Assignments Cont.

Pset 5
- Implement an analysis to check for memory errors in C

Pset 6
- Practice LTL and CTL
- Learn how to use a model checker
- Fun with synthesis