Welcome to CS106B!

- Five Handouts
  - Course information
  - Syllabus
  - Course placement information
  - Honor Code policies
  - Assignment 0: Welcome to CS106B!

- Today:
  - Course Overview
  - Where are We Going?
  - Introduction to C++
Who's Here Today?

- Aeronautics and Astronautics
- Biochemistry
- Bioengineering
- Biology
- Biomedical Informatics
- Business Administration
- Chemical Engineering
- Chemistry
- Chinese
- Civil and Environmental Engineering
- Computational and Mathematical Engineering
- Computer Science
- Creative Writing
- East Asian Studies
- Economics
- Electrical Engineering
- Energy Resources Engineering
- Engineering
- Environment and Resources
- Feminism, Gender, and Sexuality Studies
- Film and Media Studies
- German Studies
- Human Biology
- Immunology
- International Policy Studies
- Law
- Management Science and Engineering
- Materials Science and Engineering
- Mathematical and Computational Sciences
- Mechanical Engineering
- Medicine
- Music
- Petroleum Engineering
- Physics
- Political Science
- Psychology
- Public Policy
- Science, Technology, and Society
- Statistics
- Stem Cell Biology and Regenerative Medicine
- Symbolic Systems
- Theater and Performing Studies
- Undeclared!
Course Staff

**Instructor**: Keith Schwarz  
(htiek@cs.stanford.edu)

**Head TA**: Anton Apostolatos  
(antonaf@stanford.edu)

*The CS106B Section Leaders*

*The CS106B Course Helpers*
Course Website

http://cs106b.stanford.edu
Prerequisites

CS106A
(or equivalent)
(check out our course placement handout if you’re unsure!)
Required Reading

- Available in the bookstore. Some copies are on reserve in the Engineering library.
- There are (old, outdated) PDFs floating around; use them at your own risk!
- We do recommend picking up a copy of this book, since it provides a lot of useful extra background information.
Grading Policies
Grading Policies

- 35% Assignments

Eight Assignments
(One intro assignment that goes out today, seven programming assignments)
Grading Policies

35% Assignments
25% Midterm Exam
35% Final Exam
5% Section Participation

Midterm Exam
Tuesday, February 21st
7PM – 10PM
Location TBA
Grading Policies

35% Assignments
25% Midterm Exam
35% Final Exam
5% Section Participation

Final Exam
Monday, March 20th
8:30AM – 11:30AM
No alternate exams except for OAE accommodations.
Grading Policies

- 35% Assignments
- 25% Midterm Exam
- 35% Final Exam
- 5% Section Participation

Discussion Sections
Weekly sections. Let’s go talk about them!
Discussion Sections

- There are weekly discussion sections in CS106B. Section attendance is required.
- Sign up between Thursday, January 12th at 5:00PM and Sunday, January 16th at 5:00PM by visiting http://cs198.stanford.edu/section
- We don’t look at Axess for section enrollments. Please make sure to sign up here even if you’re already enrolled on Axess.
How Many Units?

```python
int numUnits(bool isGrad) {
    if (isGrad) {
        return randomInteger(3, 5); // 3 to 5
    } else {
        return 5;
    }
}
```
Getting Help
Getting Help

- **LaIR Hours!**
  - Sunday – Thursday, 6PM – Midnight
  - Starts next week.
- **Anton's Office Hours in the Huang basement**
  - Wednesdays, 2:00PM – 4:00PM
- **Keith's Office Hours in Gates 219**
  - Tuesdays, 2:15PM – 4:15PM (starting next week)
  - Come hang out and chat about whatever it is that you’re interested in!
What's Next in Computer Science?
Goals for this Course

- *Learn how to model and solve complex problems with computers.*
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.
Goals for this Course

Learn how to model and solve complex problems with computers.

To that end:

- Explore common abstractions for representing problems.

Harness recursion and understand how to think about problems recursively.

Quantitatively analyze different approaches for solving problems.
totally rocks my socks
Hey, that's us!
This structure is called a **tree**. Knowing how to model, represent, and manipulate trees in software makes it possible to solve interesting problems.
Building a vocabulary of *abstractions* makes it possible to represent and solve a wider class of problems.
How do we keep passwords secure when servers are hacked all the time?
Inputs can be just about anything: strings, ID numbers, molecular shapes, passwords, etc.

Output is a seemingly random number that serves as a "fingerprint" of the input.
Building a vocabulary of *abstractions* makes it possible to represent and solve a wider class of problems.
Goals for this Course

• *Learn how to model and solve complex problems with computers.*

• To that end:
  • Explore common abstractions for representing problems.
  • Harness recursion and understand how to think about problems recursively.
  • Quantitatively analyze different approaches for solving problems.
Goals for this Course

Learn how to model and solve complex problems with computers.

To that end:

- Explore common abstractions for representing problems.
- Harness recursion and understand how to think about problems recursively.

Quantitatively analyze different approaches for solving problems.
Creating Trees
A **recursive solution** is a solution that is defined in terms of itself.
Thinking recursively gives you a different perspective on everyday structures.
Thinking recursively allows you to model and solve an enormous class of problems cleanly and concisely.
Goals for this Course

- Learn how to model and solve complex problems with computers.

- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.
Goals for this Course

Learn how to model and solve complex problems with computers.

To that end:

- Explore common abstractions for representing problems.
- Harness recursion and understand how to think about problems recursively.
- Quantitatively analyze different approaches for solving problems.
Travel Time: $13 + 15 + 17 + 14 + 11 + 9 + 12 = 91$
Travel Time: $10 + 17 + 7 + 14 + 13 + 4 + 7 = 72$
In an $n \times n$ grid, there are at least $4^n / n$ possible paths from one corner to another.

If $n = 154$, this is approximately equal to the number of atoms in the universe.
In an $n \times n$ grid, there are at least $\frac{4^n}{n}$ possible paths from one corner to another.

If $n = 50$, it would take the lifetime of the universe to list off all possible paths.
from 0 to
from 0 to 32?

0 - 10 - 17 - 25 - 32?

0 - 13 - 6 - 22 - 40?

0 - 17 - 5 - 22 - 40?

0 - 28 - 5 - 22 - 36?

0 - 45? - 14 - 29 - 36?

from 0 to 32?

0 - 10 - 17 - 25 - 32?

0 - 13 - 6 - 22 - 40?

0 - 17 - 5 - 22 - 40?

0 - 28 - 5 - 22 - 36?

0 - 45? - 14 - 29 - 36?

from 0 to 32?

0 - 10 - 17 - 25 - 32?

0 - 13 - 6 - 22 - 40?

0 - 17 - 5 - 22 - 40?

0 - 28 - 5 - 22 - 36?

0 - 45? - 14 - 29 - 36?

from 0 to 32?
This approach is called Dijkstra's Algorithm.

For an $n \times n$ grid, it requires $(\text{roughly speaking}) n \log n$ operations to find the shortest path.
This approach is called **Dijkstra's Algorithm**.

Google Maps uses a slightly modified version of this algorithm.
This approach is called Dijkstra's Algorithm.

Google Maps uses a slightly modified version of this algorithm.

With $n$ intersections, it requires roughly $n \log n$ operations to find the shortest path.
Goals for this Course

- *Learn how to model and solve complex problems with computers.*
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.
Who's Here Today?

- Aeronautics and Astronautics
- Biochemistry
- Bioengineering
- Biology
- Biomedical Informatics
- Business Administration
- Chemical Engineering
- Chemistry
- Chinese
- Civil and Environmental Engineering
- Computational and Mathematical Engineering
- Computer Science
- Creative Writing
- East Asian Studies
- Economics
- Electrical Engineering
- Energy Resources Engineering
- Engineering
- Environment and Resources
- Feminism, Gender, and Sexuality Studies
- Film and Media Studies
- German Studies
- Human Biology
- Immunology
- International Policy Studies
- Law
- Management Science and Engineering
- Materials Science and Engineering
- Mathematical and Computational Sciences
- Mechanical Engineering
- Medicine
- Music
- Petroleum Engineering
- Physics
- Political Science
- Psychology
- Public Policy
- Science, Technology, and Society
- Statistics
- Stem Cell Biology and Regenerative Medicine
- Symbolic Systems
- Theater and Performing Studies
- Undeclared!
One more detail...
C++
What is C++?

• C++ is a widely used programming language used to design all sorts of systems that are
  • complex, but
  • need to run fast.
• The syntax of Java was influenced by the syntax of C++.
• There are many features of C++ that aren’t present in Java, and those features make it an attractive language for use in CS106B.
• C++ is a huge language that’s undergone many revisions (it was invented in 1983; most recent version is C++14) and we won’t be covering it in full depth. Take CS106L or CS110 for more!
/* File: hello-world.cpp
 *
 * A canonical Hello, world! program
 * in C++.
 */

#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
}

/* File: retain-evens.cpp
 *
 * A program to filter out odd numbers from a list.
 */

#include <iostream>
#include "vector.h"

using namespace std;

Vector<int> retainEvens(Vector<int> values) {
    Vector<int> result;
    for (int i = 0; i < values.size(); i++) {
        if (values[i] % 2 == 0)
            result += values[i];
    }
    return result;
}

int main() {
    Vector<int> values;
    values += 1, 2, 3, 4, 5;

    Vector<int> processed = retainEvens(values);

    for (int elem: processed) {
        cout << elem << endl;
    }
}
Your Action Items

- **Assignment 0: Welcome to CS106B** is due this Friday at the start of class (11:30AM).
  - Starter files and assignment handout are up on the course website.
  - No programming involved, but you’ll need to get your development environment set up.
  - There’s a bunch of documentation up on the course website. Please feel free to reach out to us if there’s anything we can do to help out!
- Some of the later assignments can be done in pairs. You may want to start thinking about who you’d like to work with, since you’ll need to register for the same section as the person you’ll be working with.
Next Time

- Welcome to C++!
  - Defining functions.
  - Reference parameters.
  - Introduction to recursion.