Programming Abstractions in C++

CS106B

Instructors:
Cynthia Bailey Lee
Julie Zelenski

Meet your instructors: Cynthia Bailey Lee

Research Interests

- UCSD PhD in large-scale computing
- Recently: computer science education, DEI in tech, justice and social impacts of tech

Teaching

- At Stanford since 2013
- Primarily CS106B and CS103 (maybe see you in Winter or Spring?)

Software engineer

- Startups, NASA, litigation consulting

Away from keyboard

- Family, biking, hiking
- “like a cat lady, but for chickens”

Meet your instructors: Julie Zelenski

Proud Stanford alum (undergrad and grad)

- FLI from CA Central Valley
- Coming to Stanford changed the arc of my life in every possible way
- Hope your experience is similarly transformative!

Software engineering

- NeXT Computer, acquired by Apple

Lecturer at Stanford

- Fantastic colleagues, awesome students
- CS department a research powerhouse AND deeply committed to education

Away from keyboard

- Outdoors with family as much as possible

Meet your Head TA: Neel Kishnani

Stanford BS '21, currently MS '23

- Major in CS, Minors in Education, Music
- I took CS106B Winter 2018!

Teaching

- SL since 2019, 2nd Head TA stint
- Here to help you succeed in this class, whatever that means for you. Email me neelk@stanford.edu with anything you need!

Software Engineering

- Most recently security engineering at Apple

Away from Keyboard

- Hobbies: saxophone, basketball, boba

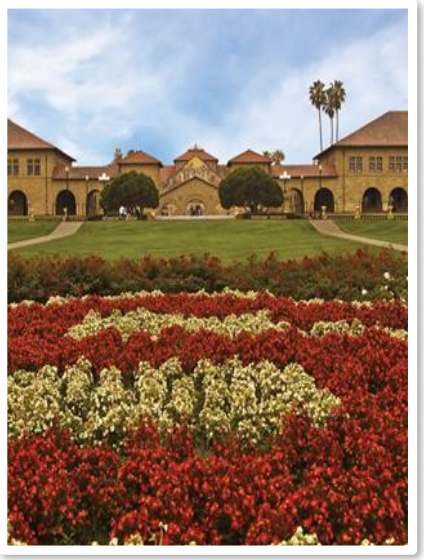
Discussion Section, Section Leaders ("SLs")

Section Leaders are helpful undergraduate assistants!

- Your personal trainer in 106B!
- (thought to keep in the back of your mind: you could be one someday…?)

Course Logistics

QUICK OVERVIEW OF HOW TO EARN THE GRADE YOU WANT IN CS106B



Course Grade Overview

Final grades for the course will be determined using the following weights:

- **55%** Programming assignments
  - Approximately weekly
- **15%** Mid-quarter exam
  - Tuesday, Nov. 1, 7-9pm
- **20%** End-quarter final exam
  - Monday, Dec. 12, 8:30-11:30am
- **5%** Section participation
  - Your chance to get a small-class experience
- **5%** Lecture participation
  - Come each day, or watch video promptly

Note: We will compute your course grade once including lecture participation and again without (moving that weight to final exam). The weighting that results in the better outcome for you is the one we will use.

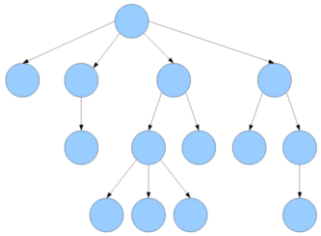
Community norms and expectations

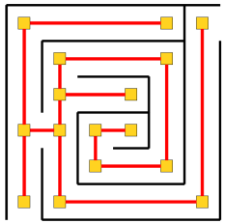
- Celebrate discovery and growth. No gatekeeping, shaming, or comparisons based on who knew what coming in.
  - *Example of things we're not going to do: audience "questions" in lecture that are just showing off that you know some jargon.*

- Shed "zero-sum" and scarcity attitudes. There are plenty of tech jobs.
  - *Others gaining strength in the power of coding doesn't take power away from you. Be helpful and encouraging, try to feel as genuinely happy when others around you succeed as when you succeed.*

- Do your own work. We take this very seriously, because that's how you grow.
  - *Nobody gets good at yoga by watching videos. You have to get on the mat, and sometimes you have to sweat. No shortcuts. We do enforce, but it can't only be about enforcement—you need to decide within yourself to hold the line on integrity.*

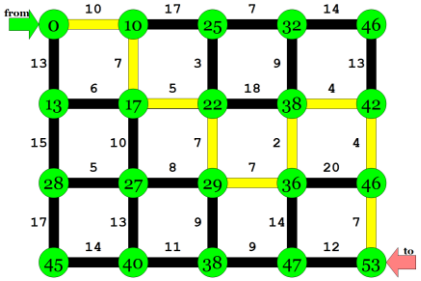
What is CS 106B?



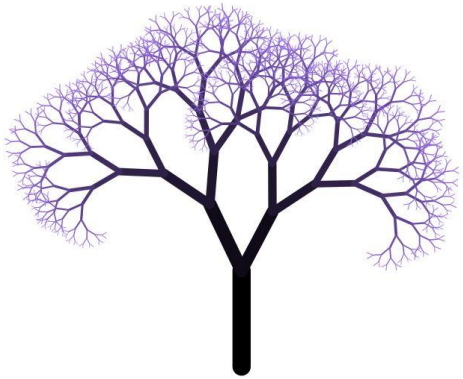


CS 106B: Programming Abstractions

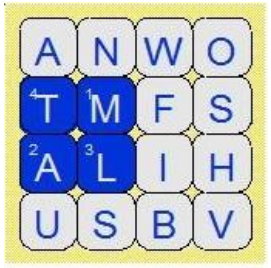
- solving big(ger) problems and processing big(ger) data
- learning to manage complex data structures
- algorithmic analysis and algorithmic techniques such as recursion
- programming style and software development practices
- familiarity with the C++ programming language



Prerequisite: CS 106A or equivalent



http://cs106b.stanford.edu/



Stanford University

CS 106L

One unit course to learn the
C++ language in depth.

Lecture: T/Th 3:00-4:20 in 380-380F
Website: http://cs106L.stanford.edu

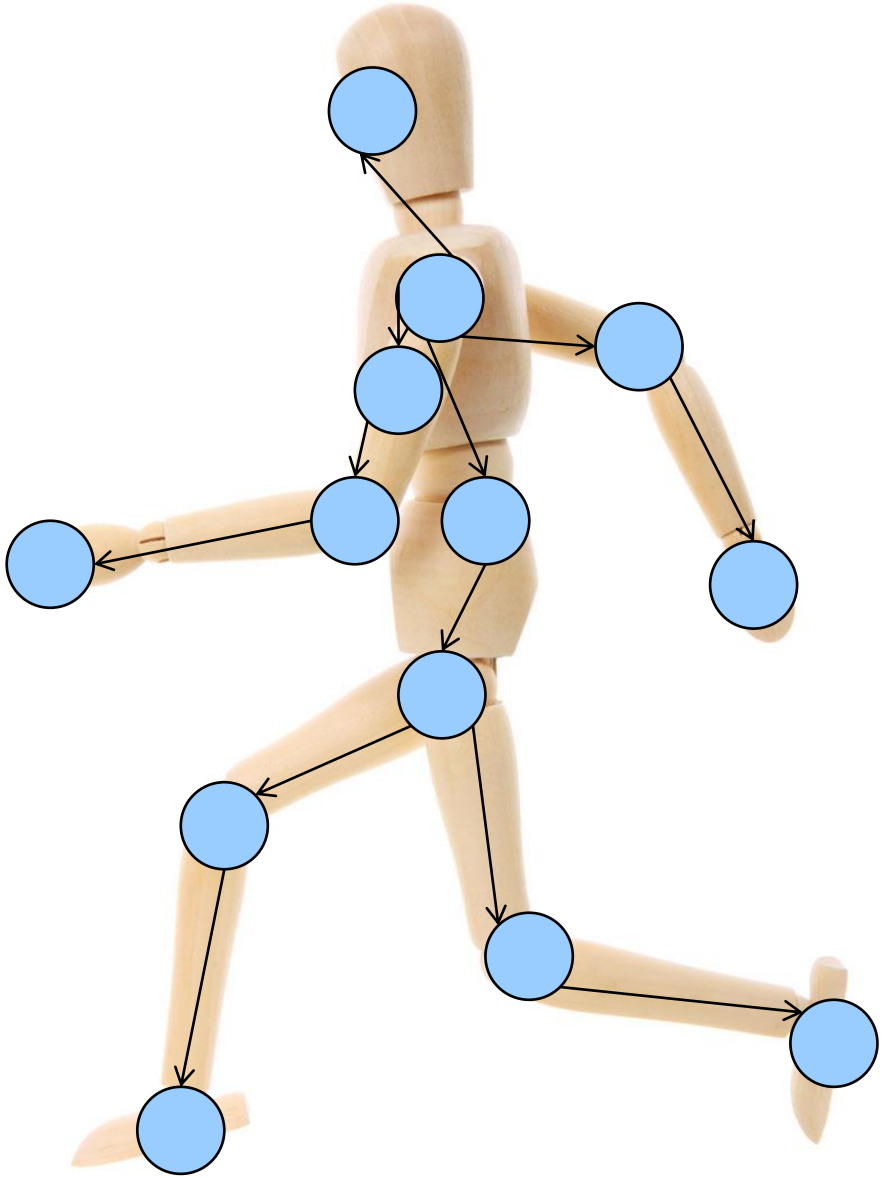
Questions?
Email us at:

Sarah McCarthy <sarahrm@stanford.edu>
Haven Whitney <havenw@stanford.edu>

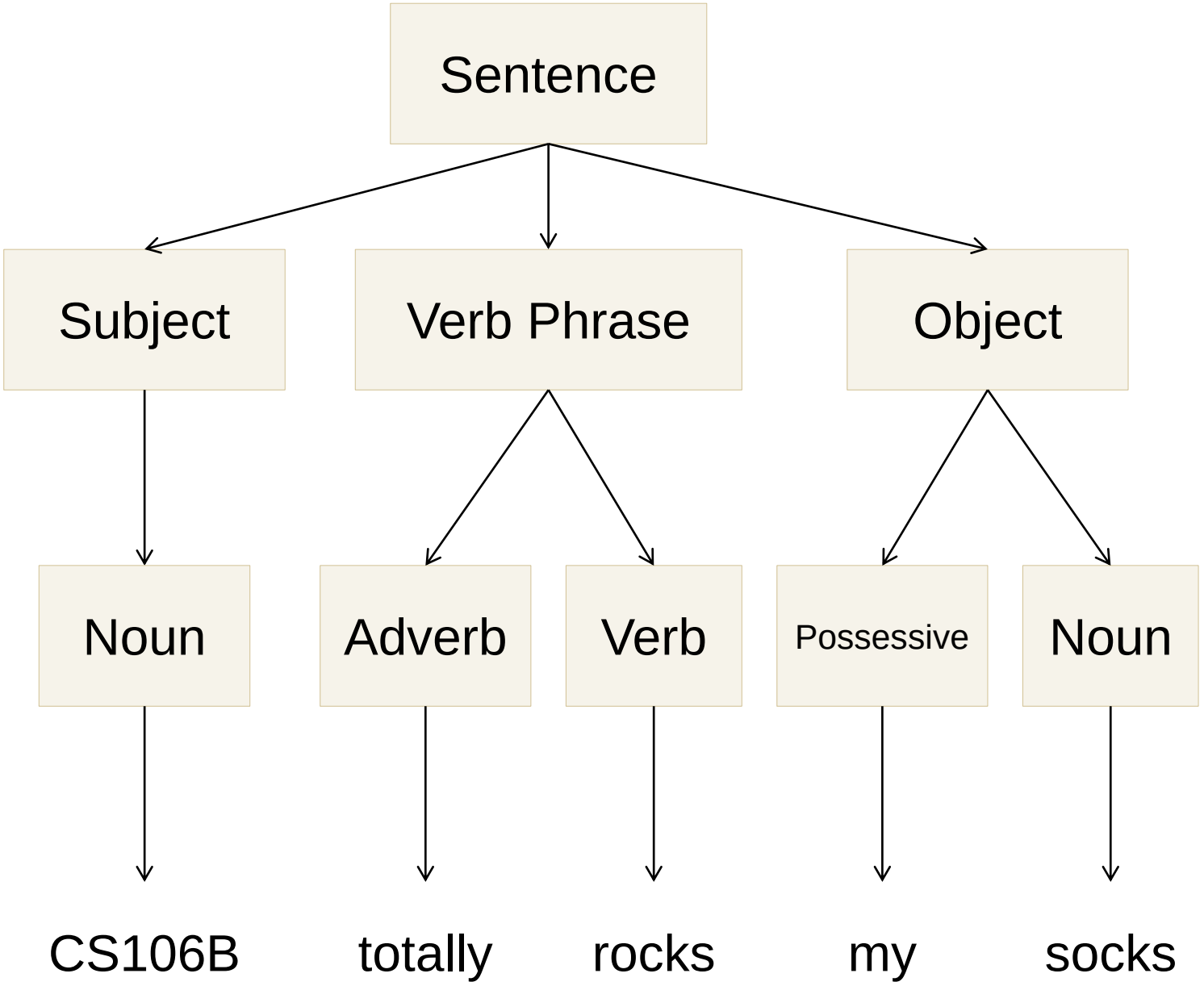
What is this class about?

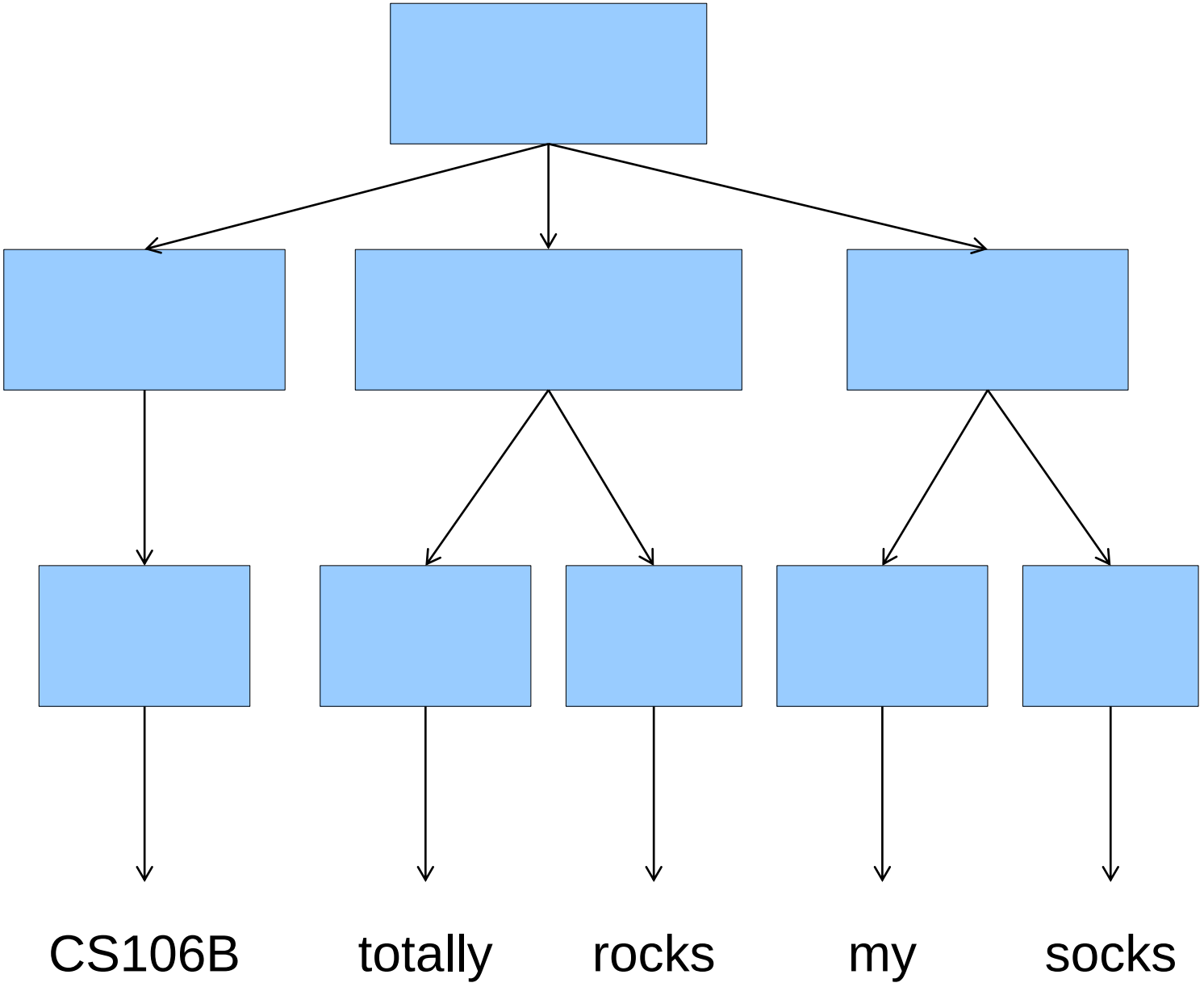
What do we mean by “abstractions”?

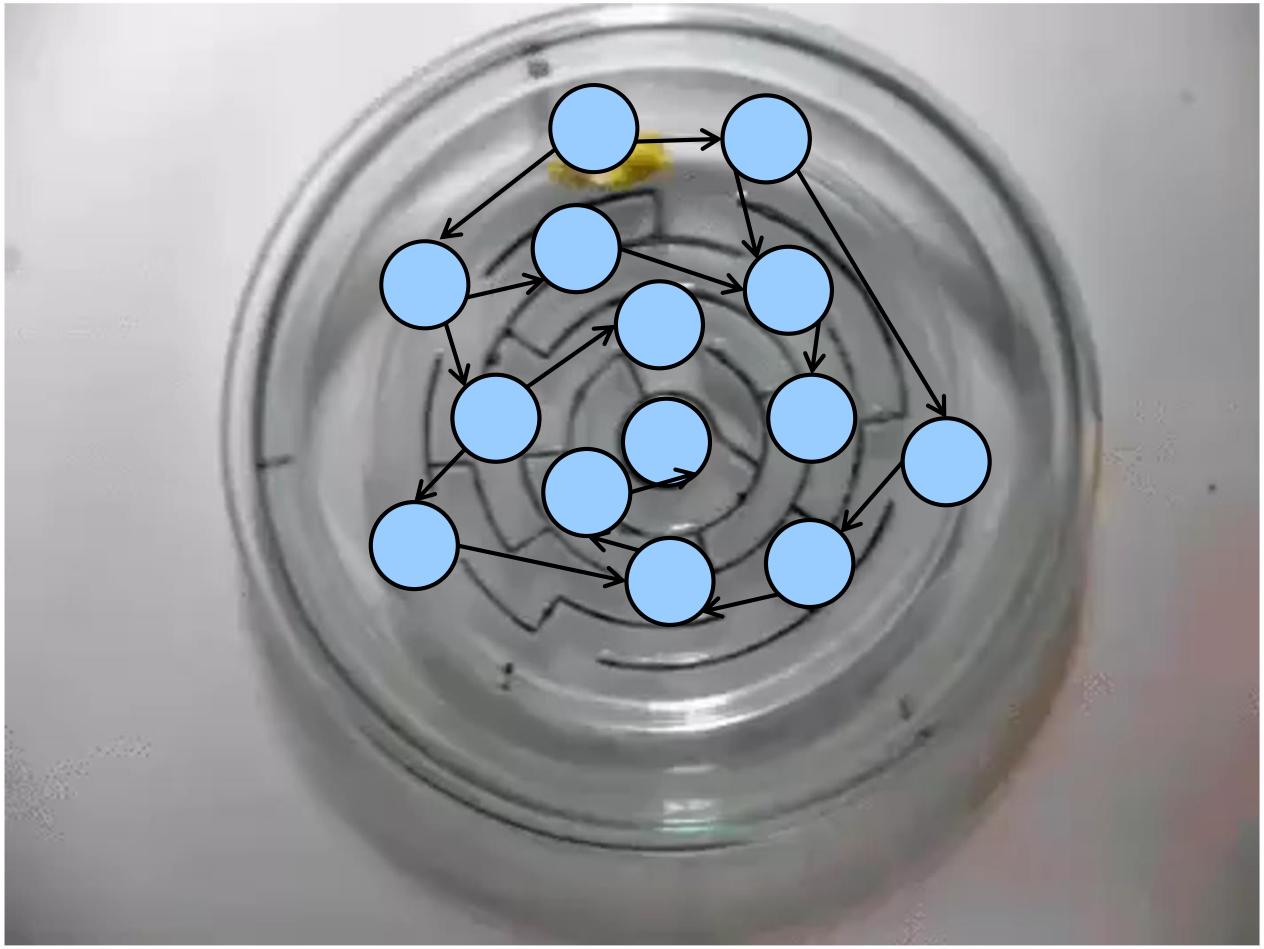
http://www.publicdomainpictures.net/pictures/10000/velka/1-1265899974oKJ9.jpg

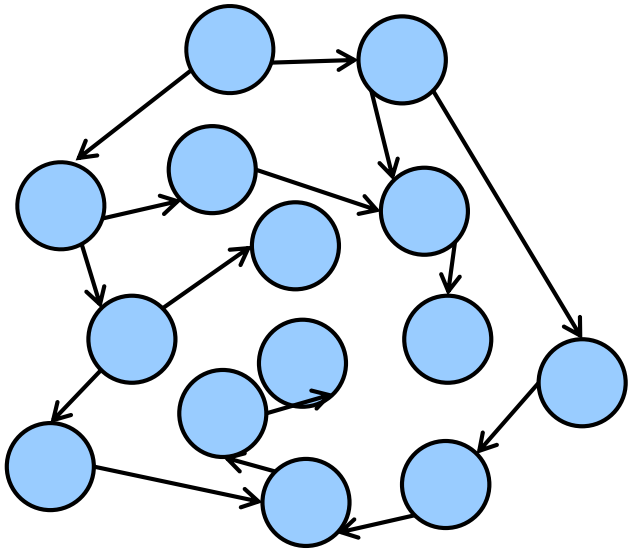


http://www.publicdomainpictures.net/pictures/10000/velka/1-1265899974oKJ9.jpg


Sentence
Subject
Verb Phrase
Object
Noun
Adverb
Verb
Possessive
Noun
CS106B
totally
rocks
my
socks

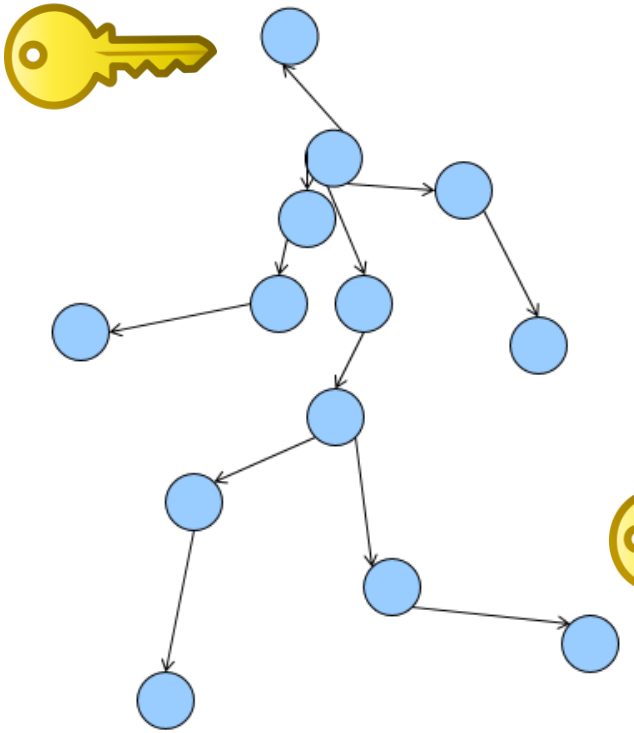

CS106B
totally
rocks
my
socks

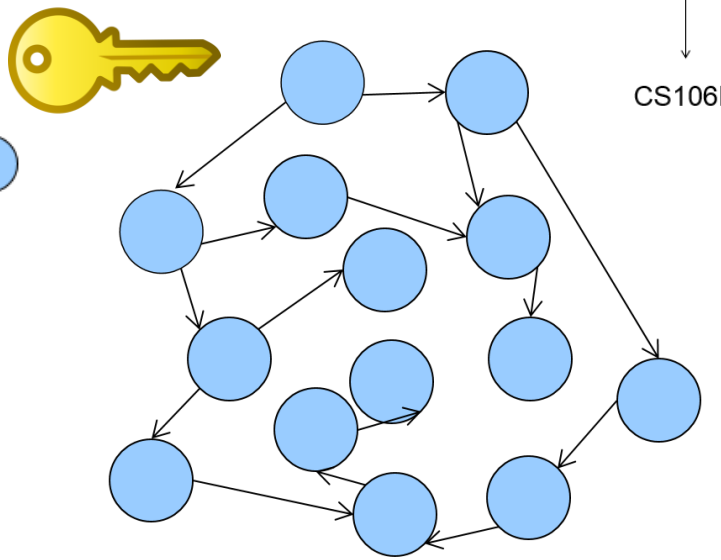


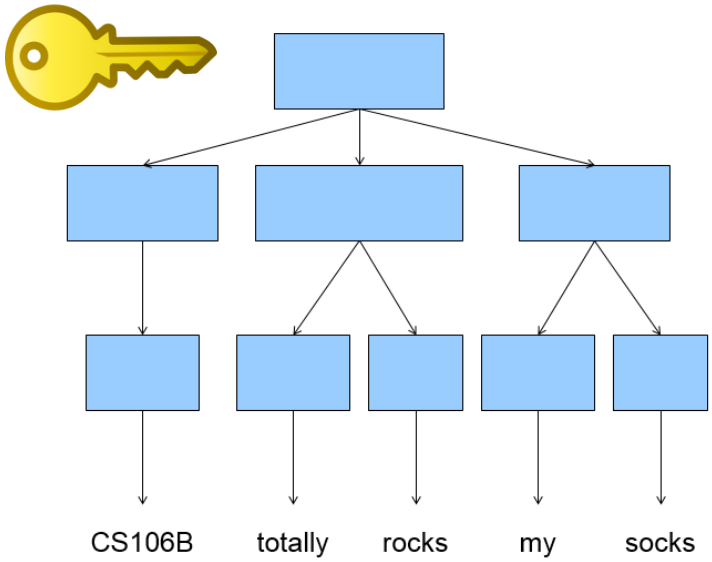


In CS106B, you'll learn to:

1. Identify common underlying structures
2. Apply known algorithmic tools that solve diverse problems that share that structure







Building a vocabulary of **abstractions**

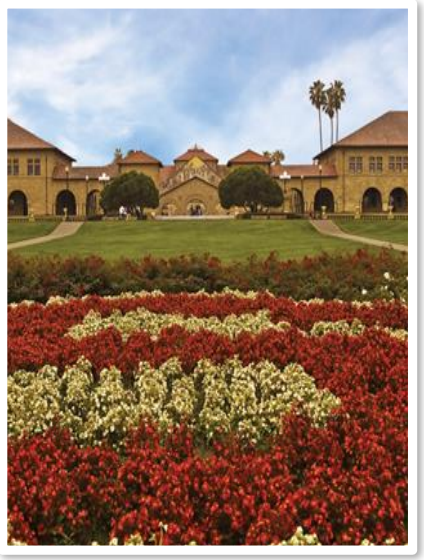
makes it possible to represent and solve a huge variety of problems using known tools.

After this course…
you'll have what is effectively a superpower.

Spend some time thinking about how you'll use it.

Welcome to C++

LET'S START CODING!!



First C++ program (1.1)

```
/*
 * hello.cpp
 * This program prints a welcome message
 * to the user.
 */
#include <iostream>
#include "console.h"
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

Include statements are like imports in Java/Python. More on this in a moment.

Every C++ program has a **main** function. The program starts at **main** and executes its statements in sequence.

At program end, **main** returns 0 to indicate successful completion. A non-zero return value is an error code, but we won't use this method of error reporting in this class so we will always return zero.

C++ variables and types (1.5-1.8)

- The C++ compiler is rather picky about *types* when it comes to variables.
- Types exist in languages like Python (see the two code examples at right), but you don't need to say much about them in the code. They just happen.
- The **first time** you introduce a variable in C++, you need to announce its type to the compiler (what kind of data it will hold).
  - After that, just use the variable name (don't repeat the type).
  - You won't be able to change the type of data later! C++ variables can only do one thing.

C++

```
int x = 42 + 7 * -5;
double pi = 3.14159;
char letter = 'Q';
bool done = true;

x = x - 3;
```

Python

```
x = 42 + 7 * -5
pi = 3.14159
letter = 'Q'
done = True

x = x - 3
```

More C++ syntax examples (1.5-1.8)

```
for (int i = 0; i < 10; i++) {      // for loops
    if (i % 2 == 0) {               // if statements
        x += i;
    }                               /* two comment styles */
}

while (letter != 'Q' && !done) {    // while loops, logic
    x = x / 2;
    if (x == 42) { return 0; }
}

binky(pi, 17);                      // function call
winky("this is a string");          // string usage
```

Some C++ logistical details (2.2)

```
#include <libraryname>    // standard C++ library
#include "libraryname.h"  // local project library
```

- Attaches a library for use in your program
- Note the differences (common bugs):
- `<>` vs `" "`
- `.h` vs no `.h`

```
using namespace name;
```

- *Mostly, just don't worry about what this actually does/means!* Copy & paste the std line below into the top of your programs.
- Brings a group of features into global scope so your program can directly refer to them
- Many C++ standard library features are in namespace std so we write:
  - `using namespace std;`
  - "std" is short for "standard"