Programming Abstractions in C++

CS106B

Cynthia Lee

Today's Topics

1. Introductions
2. Course structure and procedures
3. What is this class? What do we mean by "abstractions"?
4. Introduce the C++ language from the Java programmer's perspective (but it's ok if you're not a Java programmer!)
   - Functions
   - Strings
   - Streams

Next lecture:

- Strings and streams, continued
- Data structures: Grid

Your instructor: Cynthia Lee

RESEARCH:

- PhD @ UCSD: market-based resource allocation in large-scale systems
- Recently: computer science education

TEACHING:

- 2 years at Stanford, 3 years at UCSD
- CS106B, CS106X, CS107, CS109, CS9, SSEA (summer CS106A+B)

SOFTWARE ENGINEER:

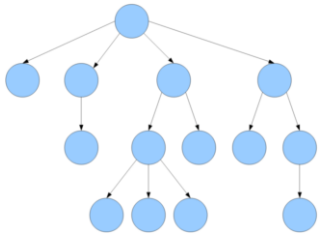
- iPhone educational games
- Document clustering and classification

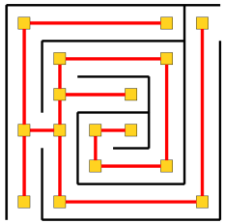
WHEN I'M NOT WORKING:

- Family, biking, climbing, hiking, pet chickens


Google

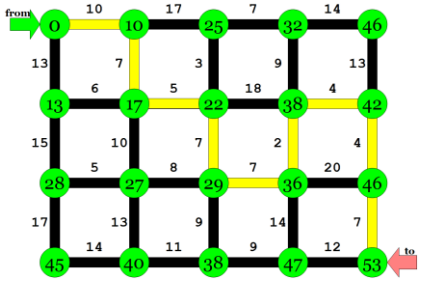
What is CS 106B?



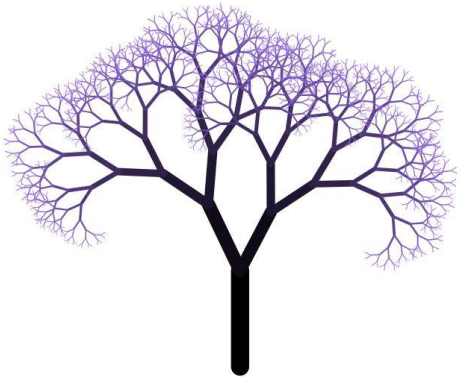


CS 106B: PROGRAMMING ABSTRACTIONS

- 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/

A one-unit course to learn and practice C++ programming in depth

Tu/Th 1:30-2:20, Littlefield 107

Take it this quarter if it fits, or it will be offered again in Autumn

(does not need to be taken in the same quarter as CS106B)

CS106L

Standard C++ Programming Laboratory

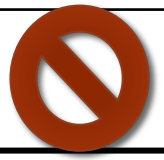
```
Base

phic clas
unctionBase() {}

 stop ction.
ec e(co t A al) ns = 0
ctionB clone() nst 0;

ived cl that executes a specific type of fun
name UnaryFunction> class FunctionImpl: public F

unctionImpl(UnaryFunction fn) : fn(fn) {}
 execute(const Arg& val) const
```

Late Days

Late day: allows you to submit a homework **<u>1 Lecture Day</u>** late

- You get 3 free (no-penalty) late days for the quarter, but only one or two can be used on a given assignment
- After your late days are used up (or for the 3rd late day), 1 bucket grade is deducted per day
- NO SUBMISSIONS are accepted after 3 days late

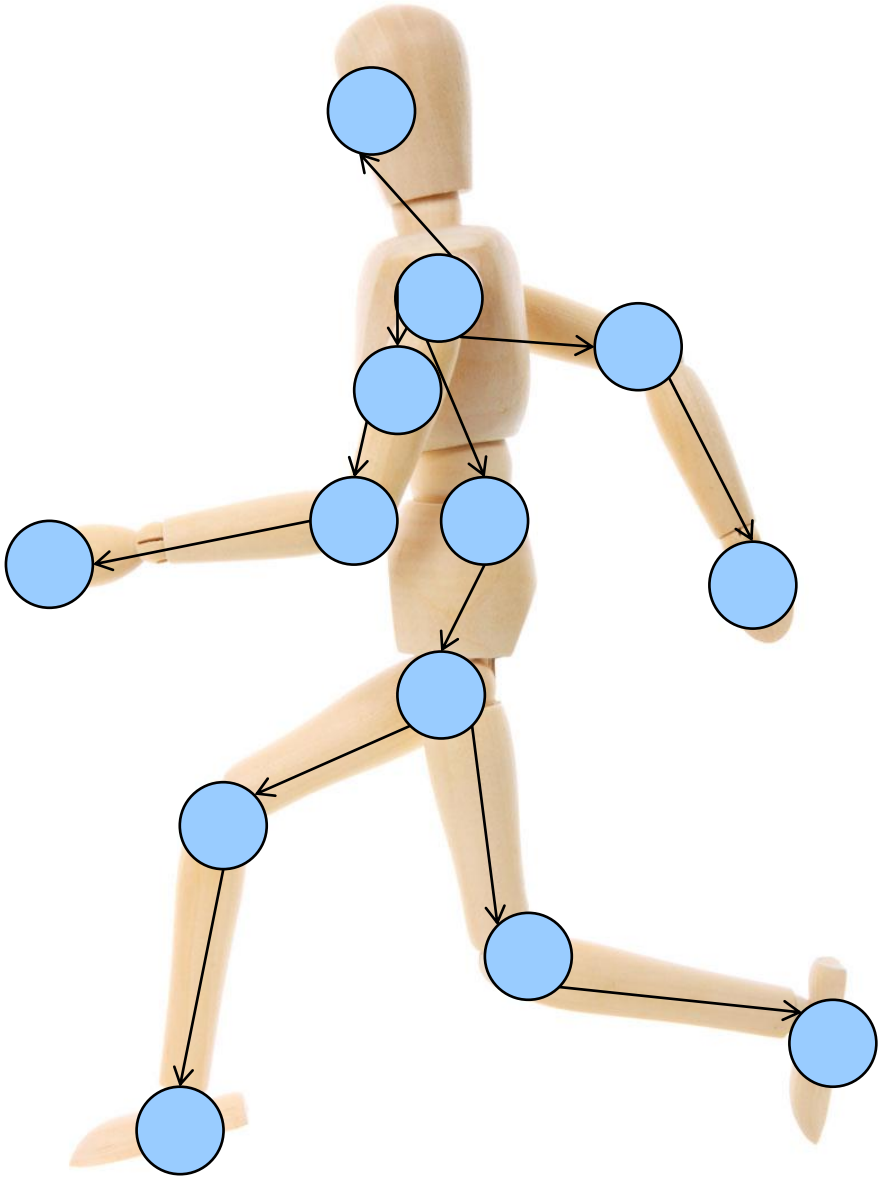
Example:

Week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1							
2				due	1 day late	1 day late	2 days late
3	2 days late	2 days late	3 days late	3 days late			

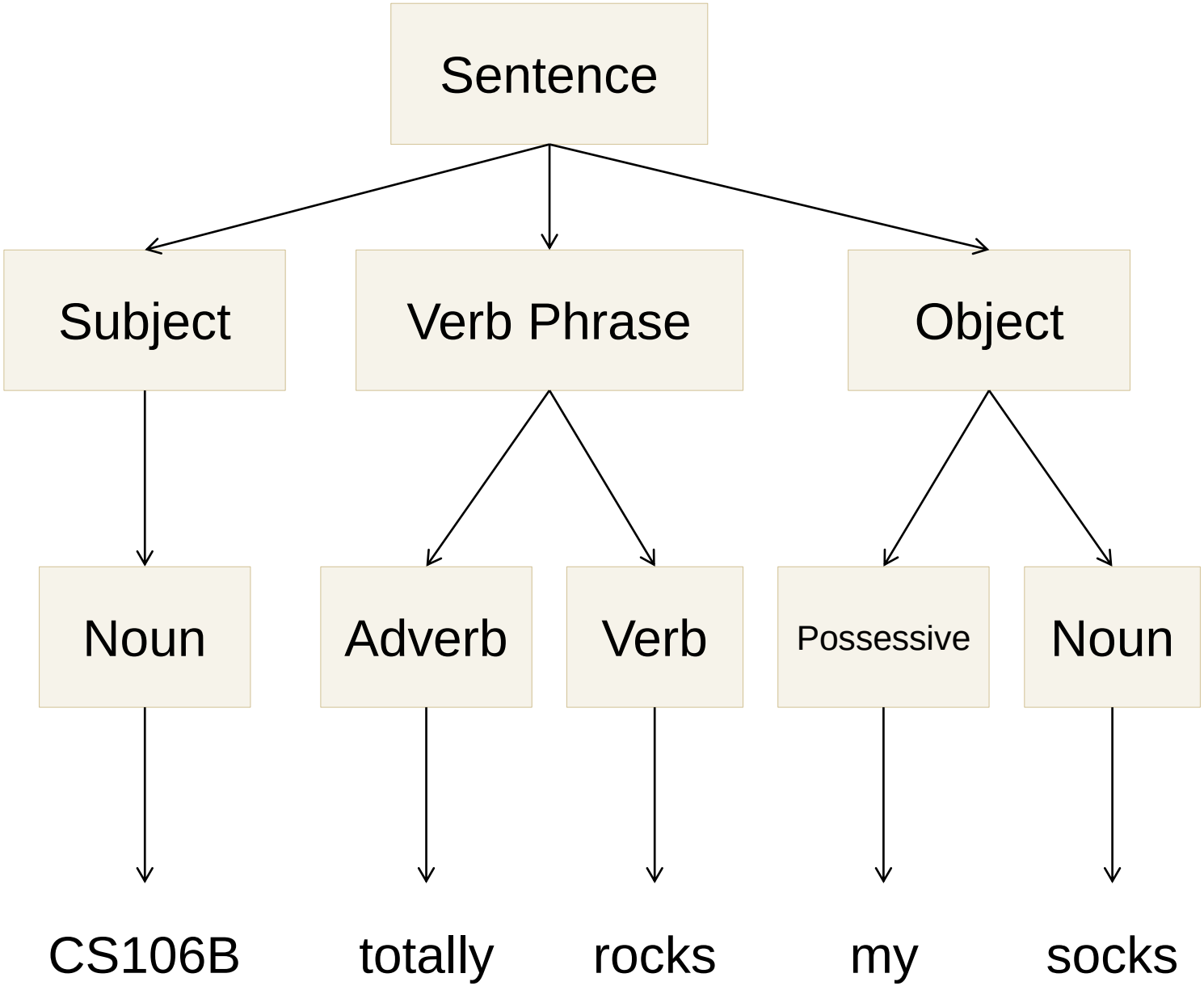
What is this class about?

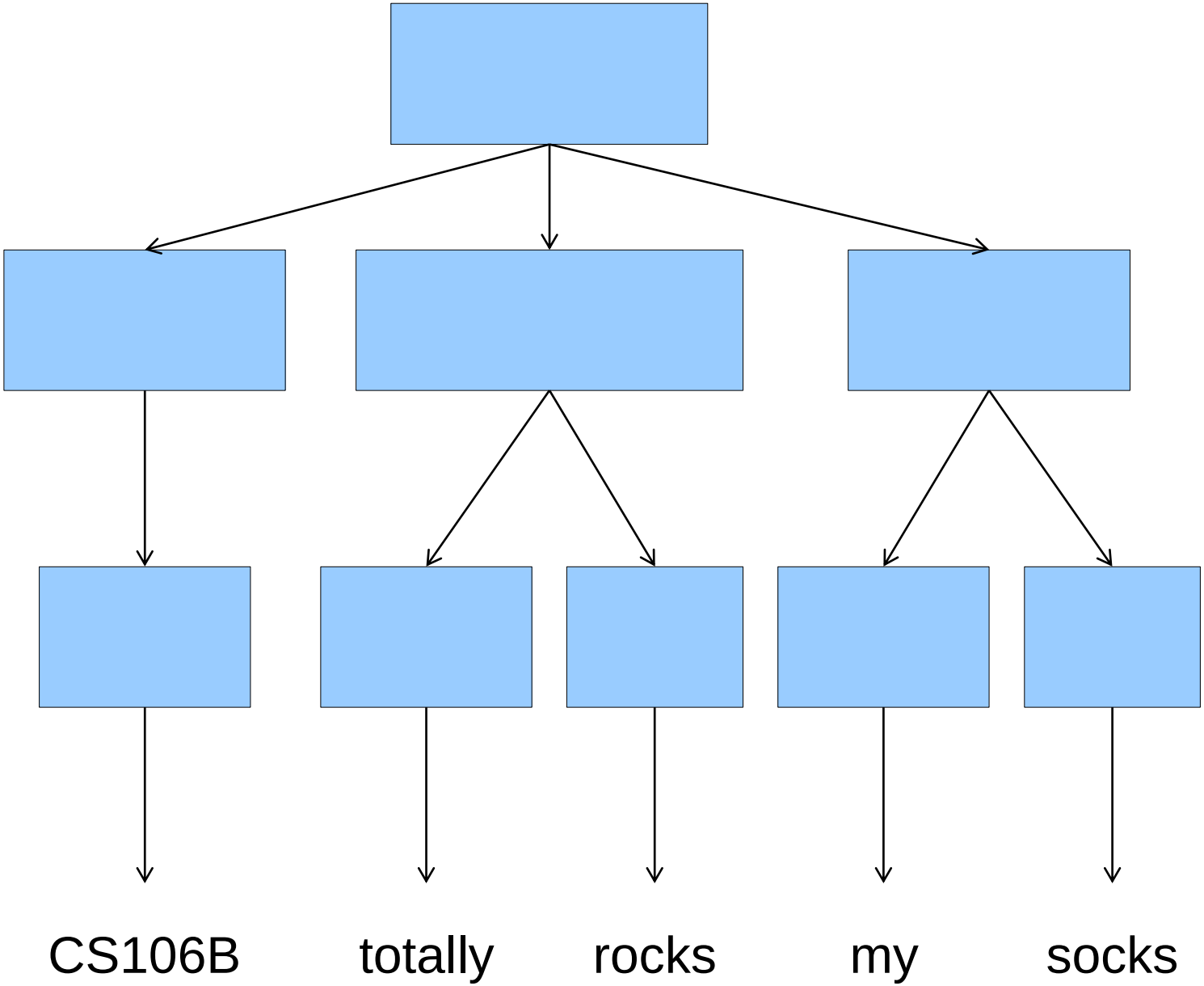
What do we mean by "abstractions"?

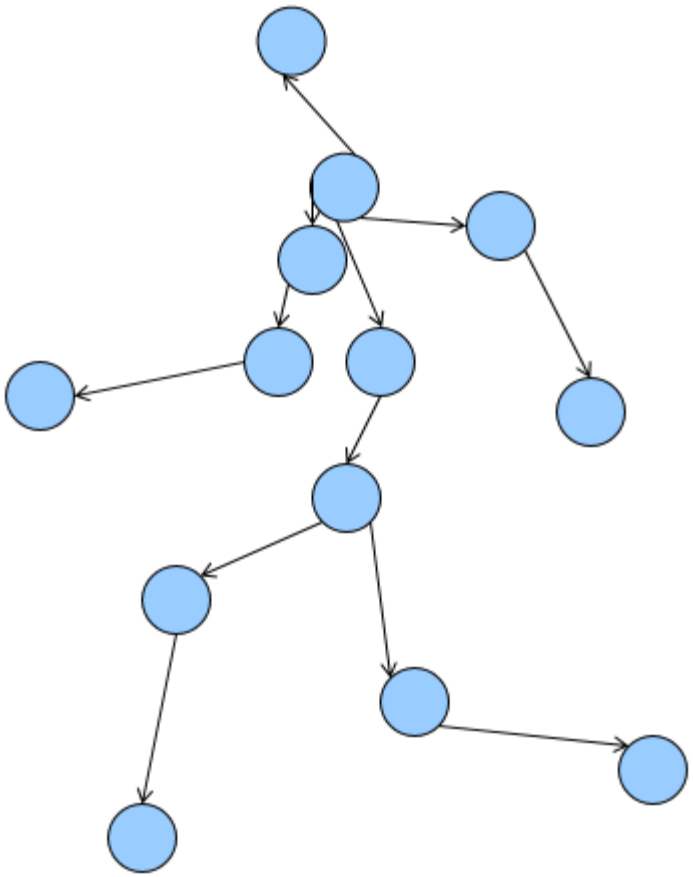
http://www.publicdomainpictures.net/pictures/10000/velka/1-1265899974oKJ9.jpg

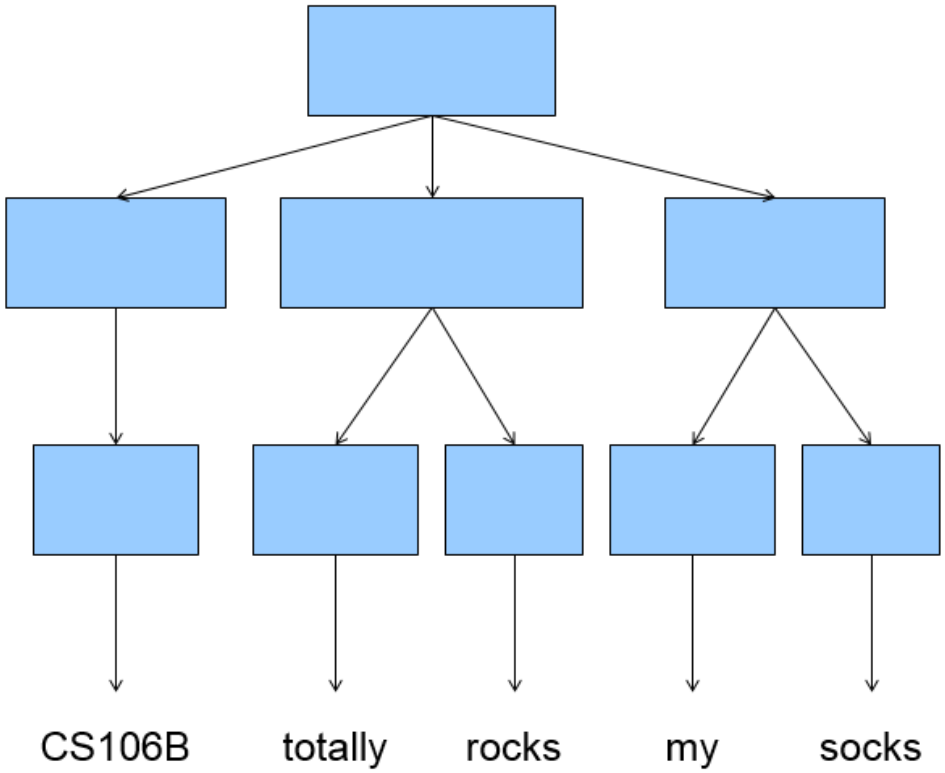


http://www.publicdomainpictures.net/pictures/10000/velka/1-1265899974oKJ9.jpg


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


CS106B
totally
rocks
my
socks




CS106B
totally
rocks
my
socks

Building a vocabulary of **abstractions**

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

A first C++ program (Error)

firstprogram.cpp

```
#include <iostream>
#include "console.h"
using namespace std;

int main(){
    cout << "|-5| = "
      << absoluteValue(-5)
      << endl;
    return 0;
}
```

```
int absoluteValue(int n) {
    if (n<0){
        return -n;
    }
    return n;
}
```

C++ from the Java Programmer's Perspective

(BUT IT'S OK IF YOU DON'T KNOW JAVA!)

A first C++ program (Error)

firstprogram.cpp

```
#include <iostream>
#include "console.h"
using namespace std;

int main(){
    cout << "|-5| = "
      << absoluteValue(-5)
      << endl;
    return 0;
}
```

```
int absoluteValue(int n) {
    if (n<0){
        return -n;
    }
    return n;
}
```

A first C++ program (Fixed #1)

firstprogram.cpp

```
#include <iostream>
#include "console.h"
using namespace std;

int absoluteValue(int n) {
    if (n<0){
        return -n;
    }
    return n;
}
```

```
int main(){
    cout << "|-5| = "
      << absoluteValue(-5)
      << endl;
    return 0;
}
```

A first C++ program (Fixed #2)

firstprogram.cpp

```
#include <iostream>
#include "console.h"
using namespace std;

int absoluteValue(int n);

int main(){
    cout << "|-5| = "
      << absoluteValue(-5)
      << endl;
    return 0;
}
```

```
int absoluteValue(int n) {
    if (n<0){
        return -n;
    }
    return n;
}
```

Design Question: Why does C++ have the function prototype syntax?

In other words, why not just have a rule that you must set up the ordering so you define your functions before using them, as in the "FIXED 1" example?

A. C++ could have done that, but such a rule would be *too cumbersome* for programmers to follow.

B. C++ could have done that, but good programming *style* dictates "top-down" approach that logically puts main() first and helper functions it calls to follow.

C. C++ could *not* have done that, because sometimes there is *no way* to order the functions so that all functions are defined before being used.

D. Other/none/more than one of the above

Design Question: Why does C++ have the function prototype syntax?

(A) and (B) The rationales behind choices (A) and (B) (previous slide) are correct

- May or may not have been enough to compel the language designers to introduce the function prototype feature

(C) is true—there are cases where you simply cannot rearrange the ordering of functions to avoid all cases of use before definition

- e.g., mutual recursion

Which came first, the chicken or the egg?

(this code is just for fun, for now—we'll cover recursion in depth in a few weeks!)

```
#include<iostream>
#include "console.h"
using namespace std;

void go(int n);
void stanford(int n);

int main(){
    int n = 5;
    go(n);
    return 0;
}
```

```
void go(int n) {
    if (n == 0) return;
    cout << "Go!" << endl;
    stanford(n-1);
}

void stanford(int n) {
    cout << "Stanford!" << endl;
    go(n);
}
```