To Ponder

Computer Science and Engineering ■ The Ohio State University

Does a problem get *easier* or *harder* to solve if I give you *less* information?

Computer Science & Engineering

Computer Science and Engineering ■ College of Engineering ■ The Ohio State University

An Introduction
(and some advanced concepts too!)

Prof. Paul Sivilotti
`sivilotti.1@osu.edu`

Where is Engineering?

Computer Science and Engineering ■ The Ohio State University

Where is Computer Science?

Computer Science and Engineering ■ The Ohio State University

Computer Science is Also…

Computer Science and Engineering ■ The Ohio State University

The First Computer Scientist

Computer Science and Engineering ■ The Ohio State University

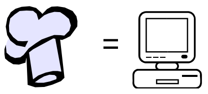
Ada Byron King,
Countess of Lovelace
1815-1852

Computers and Programs

- Computer: a device that "computes"
  - Takes inputs, produces output
- Program: a sequence of instructions
  - How to produce the output
- Contrast
  - Computers: smaller, faster, cheaper
  - Programs: larger and more complicated!

Now We're Cooking!

- Chef = computer
- Recipe = program



1. Preheat oven to 350º
2. Sift together flour, cocoa, baking powder, salt
3. Melt 1/2c butter and 1lb chocolate
4. Stir 1/2c sugar into chocolate mixture
5. Stir in 3 large eggs
6. Stir in dry ingredients
7. Add chocolate chunks
8. Form into rounded balls (1T each)
9. Bake 10 min

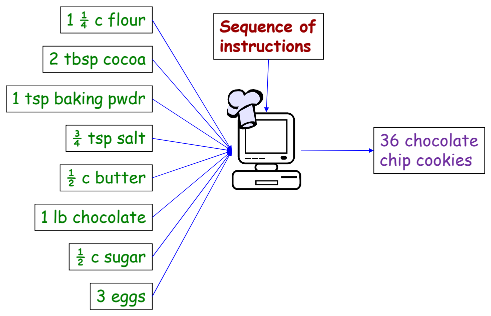
=

```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author john doe <doe.j@example.com>
 */
HelloButton()
{
  JButton hello = new JButton( "Hello, wor
  hello.addActionListener( new HelloBtnList

  // use the JFrame type until support for t
  // new component is finished
  JFrame frame = new JFrame( "Hello Button"
  Container pane = frame.getContentPane();
  pane.add( hello );
  frame.pack();
  frame.show();          // display the fra
}
```

What is the output?

Computing Choc. Chip Cookies



Requirements in Engineering

- Engineering is about problem solving
  - Given a set of requirements
  - Design a good solution
- If a design *does not* meet requirements
  - Not useful (in this case)
  - Wrong, broken, dangerous…
- Many designs *do* meet requirements
  - Which to choose? A "good" one, of course!
  - Optimization

Requirements: Example

- Span at least 9000'
- Support 6 lanes of traffic, 40 million car crossings per year
- Height at least 220'
- Withstand winds up to 50mph

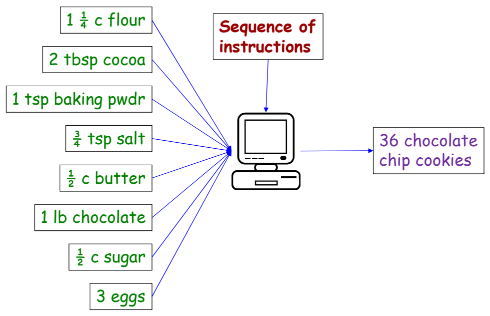
Requirements: Example #2

- Span at least 33'
- Support 2 lanes of pedestrian traffic
- Clearance at least 50'
- Prevent prisoners from escaping during crossing

Back to Software Engineering

- A software engineer builds *programs*
  - Instructions for how to turn inputs into outputs
  - Recipe engineering!
- A program must meet certain *requirements…*
  - How are requirements given for a program?
  - How are requirements given for a recipe?
  - (For software, “requirements” are usually called “specifications”)

Specifying Choc. Chip Cookies



Requirements in Software

- A software engineer builds *programs*
  - Instructions for how to turn inputs into outputs
  - Recipe engineering!
- Programs must meet *specifications*
  - What transformation to do (*not* how to do it)
  - input: ingredients
  - output: final dish
- For the same requirements, many solutions
  - Good recipes are *efficient*
  - Good recipes are *fast*
  - Good recipes are *easy to understand*
  - Good recipes are *easy to change*

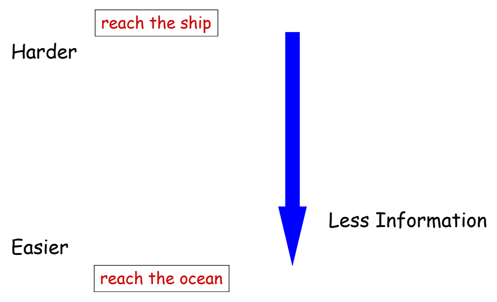
Your Turn

- Lab 1
  - You are given several specifications
  - Write programs that meet these specifications
- The best dishes are made from scratch…

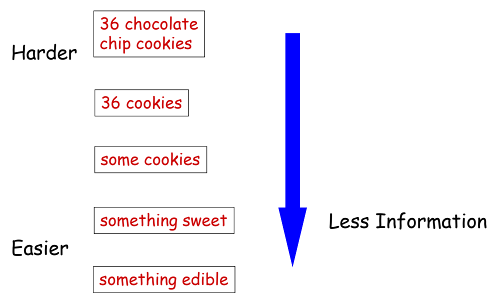
Lab 1: Debrief

- Put the problems in increasing order of difficulty:
  - A. Fixed Start / Reach the Ocean
  - B. Random-Facing Start / Reach the Ocean
  - C. Fixed-Start / Reach the Ship
  - D. All-Random Start / Reach the Ocean
- Why is C harder than A?

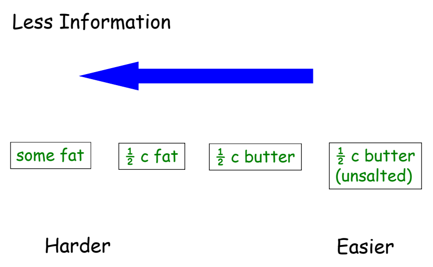
Description of Outputs



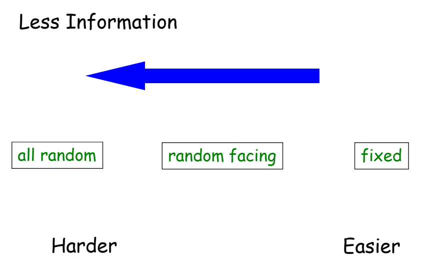
Description of Outputs



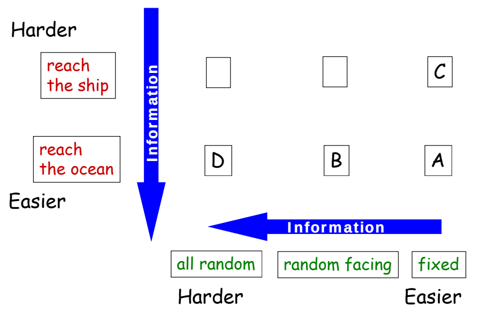
Description of Inputs



Description of Inputs



Comparing Specifications



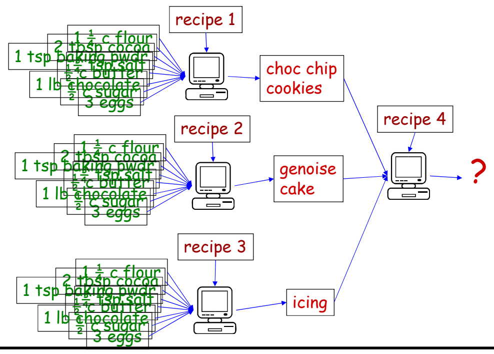
Lab 1 Take-Home Messages

- A specification that says *less* about outputs is *easier* to implement
  - But may be less useful (might not produce an appealing final dish)
- A specification that says *less* about inputs is *harder* to implement
  - But may be more useful (more general since it can be applied in more situations)

Lab 2: Composition

- Big programs are always built out of lots of smaller ones
- Output from one program can be used as input to another
- Example
  - recipe for chocolate chip cookies
  - recipe for chocolate genoise cake
  - recipe for frosting

Building a Big Recipe



Take-Home Messages

- Computer program: a sequence of instructions
  - A recipe for a chef
- Specifications: what to do (not how)
  - Given in terms of inputs and outputs
  - Less information about outputs, easier to implement
  - Less information about inputs, harder to implement
- Software engineering: how to design programs
  - Recipe design: correct, easy to understand and modify
  - Usually work in teams: communication & coordination
- Composition: big programs from smaller ones
  - Output of one program can be input to another

Computer Science & Engineering

An Introduction

(and some advanced concepts too!)

Prof. Paul Sivilotti
sivilotti.1@osu.edu