CS 161
Introduction to CS I

- How do computers make decisions?
- How can we guard against user mistakes?
- Introduce Assignment 2

But first… Best worst advice from Lab 1!

- Coursework
  - “Don't do any of your homework. Ever. It is a conspiracy theory.”
- Lecture
  - “Never go to class. It just takes away time from doing productive things like partying.”
- Programming
  - ”Make sure you only save your code after you're done.”
- Integrity
  - “Take all of your code from online solutions. Originality is for chumps.”
- Social advice
  - "Don't make friends in freshman year. You will DEFINITELY not regret it.”

Oregon State University
College of Engineering

Laura Jiang: ACM-W

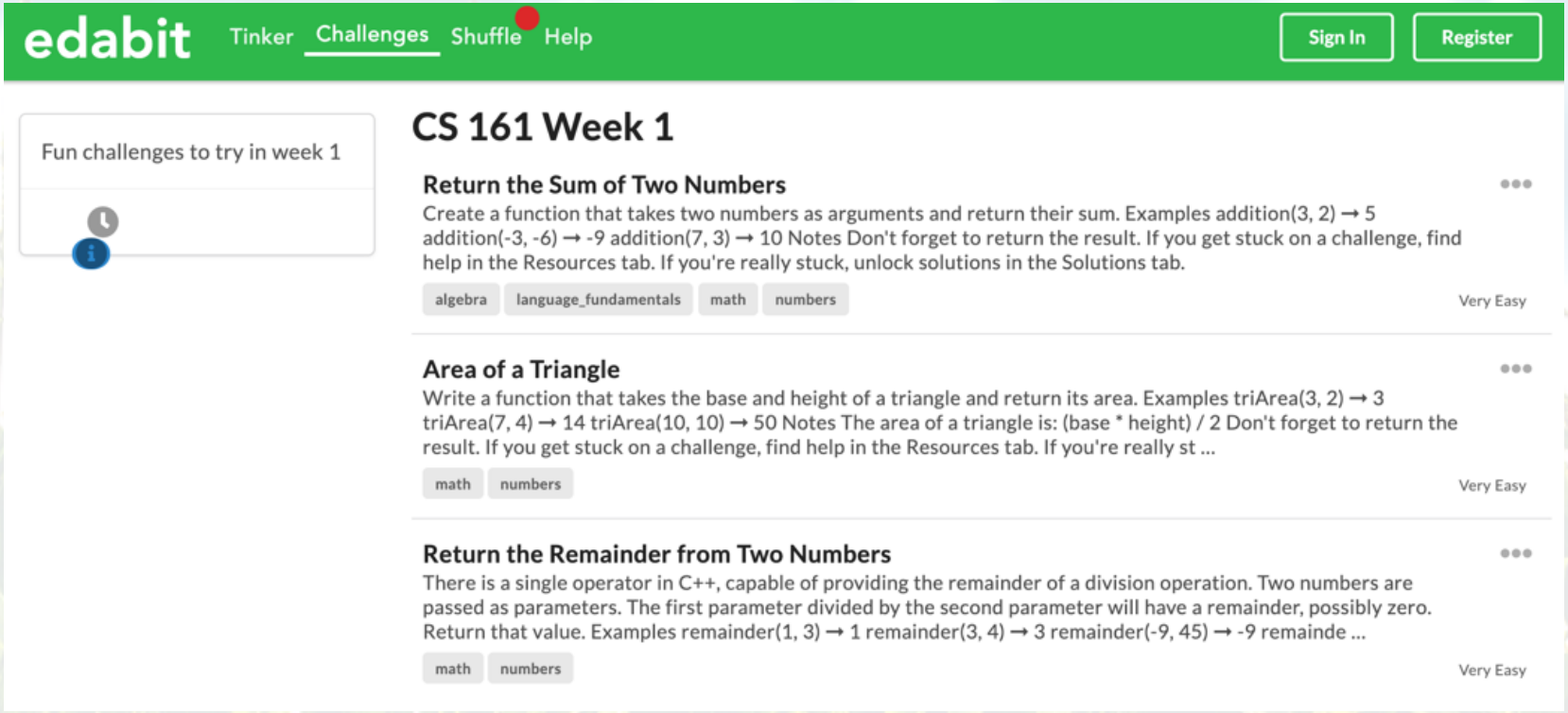
Oregon State University
College of Engineering

What's happening this week?

- http://classes.engr.oregonstate.edu/eecs/winter2020/cs161-020/calendar/
  - Lab 2 – More linux, data types, random numbers, and if/then
  - Assignment 2 Design – due 1/19
  - Week 3: Assignment 2 Design Peer Review – due 1/22
  - Week 3: Assignment 2 – due 1/26
- Great job answering each others' questions on Piazza!
  - I will give extra credit (towards exam grade) for top contributors (with answers endorsed by instructor/TA)
- Have you tried Edabit? Is it useful?

Oregon State University
College of Engineering

Edabit: Fun programming practice



Minimum and maximum values

Type	Minimum	Maximum
short	-32,768	+32,767
unsigned short	0	65,535
int	-2,147,483,648	+2,147,483,647
unsigned int	0	4,294,967,295
long	-9,223,372,036,854,775,808	+9,223,372,036,854,775,807
unsigned long	0	18,446,744,073,709,551,615
float	1.2e-38	3.4e38
double	2.2e-308	1.8e308

Minimum magnitude, not minimum (negative) value

Minimum and maximum values

Type	Minimum	Maximum
short	-32,768	+32,767
unsigned short	0	65,535
int	-2,147,483,648	+2,147,483,647
unsigned int	0	4,294,967,295
long	-9,223,372,036,854,775,808	+9,223,372,036,854,775,807
unsigned long	0	18,446,744,073,709,551,615
float	-3.4e38	3.4e38
double	-1.8e308	1.8e308

Choosing a data type

- Whole numbers:
  - (1) Allow positive and negative (`signed`) or positive only (`unsigned`) ?
  - (2) Range of values? Pick `short` or `int` or `long`
    - Think “small”, “medium”, “large”
- Real numbers: `float` or `double`
  - Think “small”, “large”
- It’s okay if you haven’t mastered this yet
  - This is a skill you will continue to practice in every program you write

Expressions

- Increment/decrement:

Correction:

- `x = 99;  x++ + 1` => 100 (and x is now 100)
- `x = 99;  x + 1` => 100 (and x is still 99)
- Here, x++ uses **postfix** notation, so the increment happens after evaluation. Compare with the **prefix** increment operator:
- `x = 99;  ++x + 1` => 101 (and x is now 100)
- Increment/decrement are for variables only (cannot do `3++`)

- Other useful expressions: **compound assignment**
  - `height = height + 3;` is the same as `height += 3;`
  - `score = score - 5;` is the same as `score -= 5;`
  - `miles = miles / 5;` is the same as `miles /= 5;`

Type casting

- Integer arithmetic
  - `6/3` => 2
  - `3/2` => 1 (!)
- Floating point arithmetic
  - `3.0/2.0` = 1.5

- You can **cast** a variable to a different type
  - `(float)3/(float)2` => ?
  - `(float)(3/2)` => ?
  - `(int)3.14159` => ?
  - `(short)3.14159` => ?
  - `(short)50000` => ?
  - `(unsigned short)50000` => ?

Course map

Divide and conquer part 2
(recursion)

Structured data
(arrays)

Basics
Storing data, calculations, interacting with users

Decision making (adaptation) and **repetition** (write once, repeat forever!)

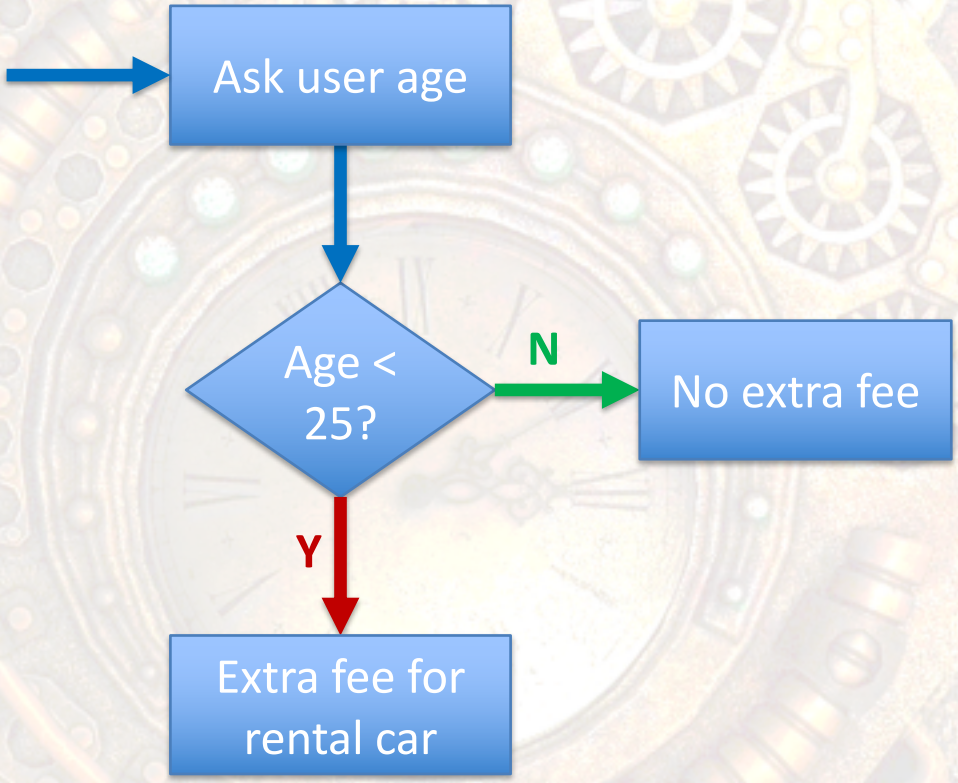
Divide and conquer
(modularization and code re-use in functions)

Dynamic growth
(memory allocation and management)

Decision making

- So far, our programs do the same thing every time.
- Life is more complicated than that!
- Often programs need to adapt or change based on user input or other variables

Flowchart for decisions



If/then/else statement in C++:

```
/* If user is under 25,
 * they must pay an extra fee */
if (age < 25)
  {
    cout << "You must pay an extra fee "
         << "to rent this car." << endl;
  }
else
  {
    cout << "No extra fee for you!" << endl;
  }
```

Boolean expressions

- **Boolean** expressions evaluate to `true` or `false`

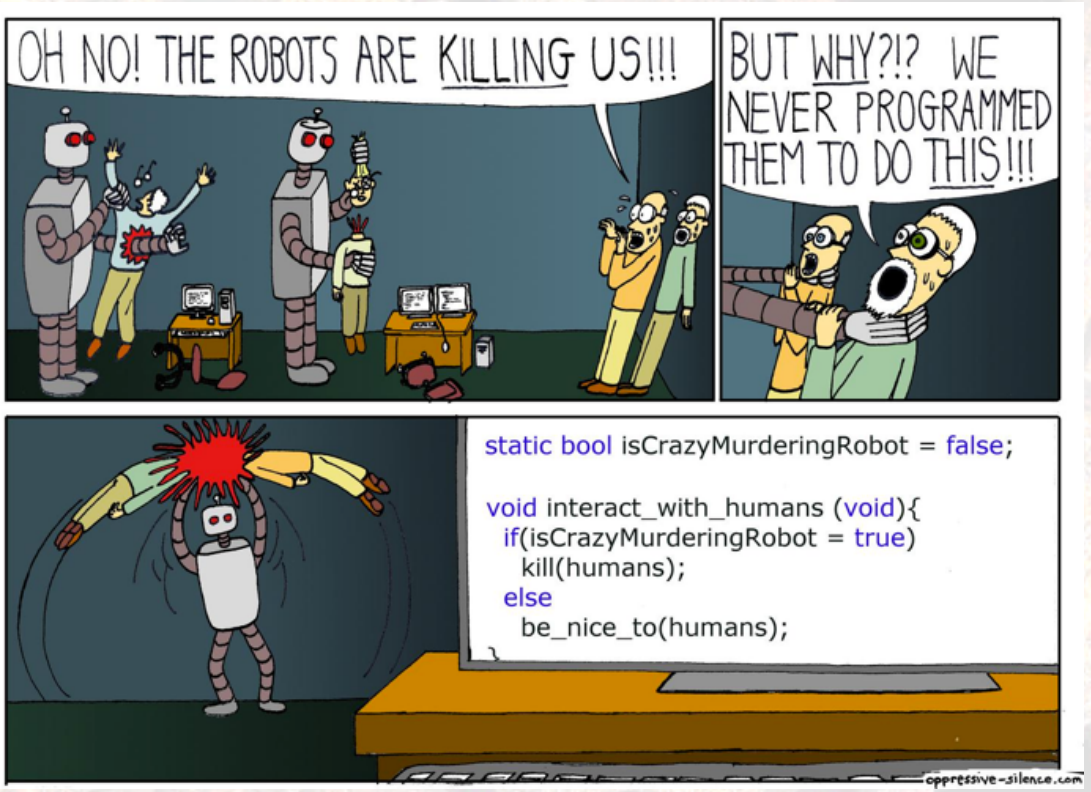
Relational operators
- `<` less than
- `>` greater than
- `<=` less than or equal
- `>=` greater than or equal
- `==` equal to
- `!=` not equal to

- Numeric values: 0 => `false`; other than 0 => `true`

- Examples
  - 3 == 3
  - -3
  - 5 > 3
  - 2 != 2
  - 2 + 1
  - 3 – 3
  - 4.1 != 4
  - 3 + 2 * 2 > 9

Programming tips

- Common mistake:
  Using = instead of ==



http://oppressive-silence.com/comic/oh-no-the-robots

Programming tips

- Common mistake:
  Using = instead of ==
- Common mistake:

```
if (n_heads == 1);
{ …
}
```

- Good practice: always have an `else` clause. Why?

Assignment 2 – Text Adventure

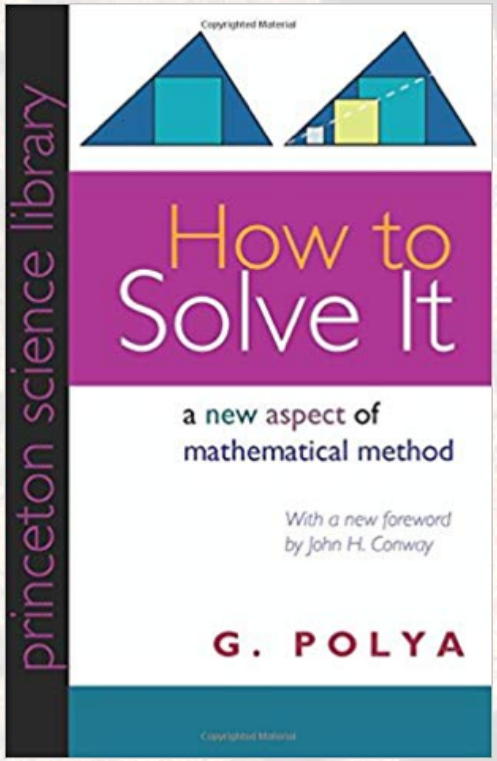
- + Design
- + Peer review

```
wkiri@madrone demos % ./assign2_game
Welcome to Mythago Wood!
 You have 0 points.

You are in a forest and see a cottage.  Do you:
(1) Knock on the door, or
(2) Keep walking?
```

Polya: "How to Solve It"

- (1) Understand the problem
  - What is the goal?
  - What are my assumptions?
- (2) Devise a plan
  - What are the steps needed?
- (3) Carry out the plan
- (4) Looking back
  - Did it work?
  - What would be an improvement?



CS 161 Design Document

- (1) Understand the problem
  - What is the goal?
  - What are my assumptions?
- (2) Devise a plan
  - What are the steps needed?
    - Pseudocode (not C++) or
    - Flowchart (see Rao p. 114)
- (4) Test plan
  - Brainstorm all possible inputs (test cases)
  - What is the expected output for each one?

What vocabulary did we learn today?

- Cast (data types)
- Compound assignment: += and -=
- Boolean expression (true/false)
- Relational operators: <, >, <=, >=, ==, !=
- Conditional statement: if/then/else

Oregon State University
College of Engineering

What ideas and skills did we learn today?

- Branching based on expression evaluation
- Check that assumptions about user input are met
- The power of `else`
- Design
  - Understand the problem + assumptions
  - Plan a solution
  - Develop tests before writing the program

Week 2 begins!

- ❑ Attend lab (laptop required)
- ❑ Read **Rao Lesson 5** (pp. 92-100)
  and **Rao Lesson 6** (pp. 113-127)
- ❑ Look at **Assignment 2** and
  plan your **design** (due **Sunday, Jan. 19**)

See you on Wednesday!

- ❑ Bring: your questions about Assignment 2