Objectives

- Concluding CS111
  - Other programming languages
  - What is computer science?
- Broader Issue: DARPA Urban Challenge

Applying What You Know To Other Languages

- At the beginning of the semester, some of you asked:
  - "Why Python?"
  - "Will I be able to read/write programs in other programming languages?"
- We'll answer the first by showing that you can do the second

Applying What You Know To Other Languages

- **Syntax**: symbols used
- **Semantics**: what the symbols *mean*

What is the Python Program doing?

What is the Python Program doing?

- Getting a line of input from "standard in" (from the user)
- Splitting the input into integers
- Calculating a formula
- Deciding if a student is admitted, based on the result of the formula

Admissions Problem

- Binary University decides to admit students based on a formula, weighting various factors
  - Scores of 70 or better are admitted
- Input: single line, 4 integers, in order below

Category	Range	Weight Factor (Multiplier)
High School GPA	0-100	0.25
SAT Score	600-2400	0.01
AP Courses	0-10	10
Intangibles	1-10	8

What is the Python Program doing?

- Getting a line of input from “standard in” (from the user)
- Splitting the input into integers
- Calculating a formula
- Deciding if a student is admitted, based on the result of the formula

Identify these pieces in the other programs

Example Programs

- printLab.sh
  - Bash script
- Java, C++, C
  - Programming contest problem: determining if someone should be admitted to college

Comparing Programming Languages

- How is the syntax/semantics of these languages different from Python?
- What is easier or harder to do in these other programming languages than in Python?

Comparing Programming Languages

- Benefits of Python:
  - Simpler syntax
  - Can cover some content with less overhead
- Drawbacks
  - Data types aren’t explicit (static)
    - Can be harder for you to remember and keep straight
  - Not compiled explicitly beforehand
    - Keep executing to find all the syntax bugs
  - Allows you to do some crazy stuff that won’t work in other programming languages

Computer Science != Programming

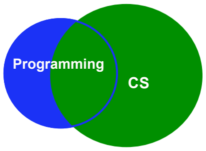
programming : CS ::

machining : engineering

grammar : literature

equations : mathematics

walking : W&L



a vehicle, not a destination

Computer Science Fields

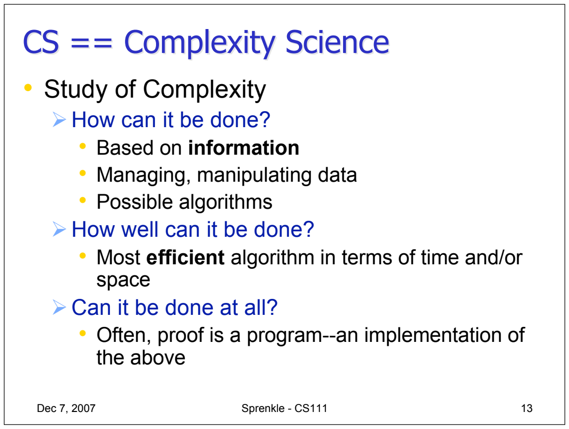
Systems
- Architecture*
- Operating systems*
- Networks*
- Distributed* and parallel systems
- Databases
- …

Software
- Compilers*
- Graphics*
- Software engineering*
- Software testing* and verification
- …

Theory
- Algorithms*
- Theory of computation
- …

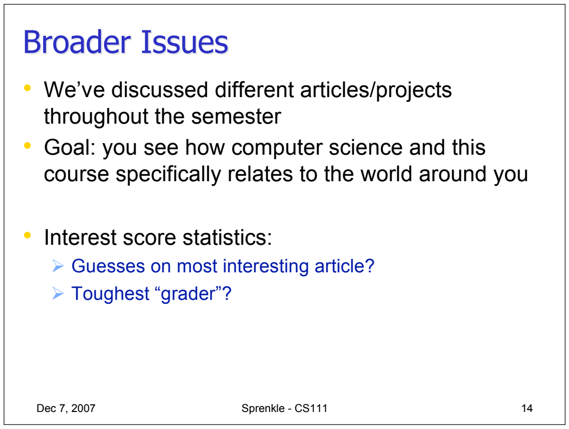
Other
- Artificial intelligence*
- Robotics
- Natural language processing*
- Bioinformatics
- Visualization*
- Numerical analysis
- …

- * = field we discussed or did a problem in
  - Some are a stretch :)



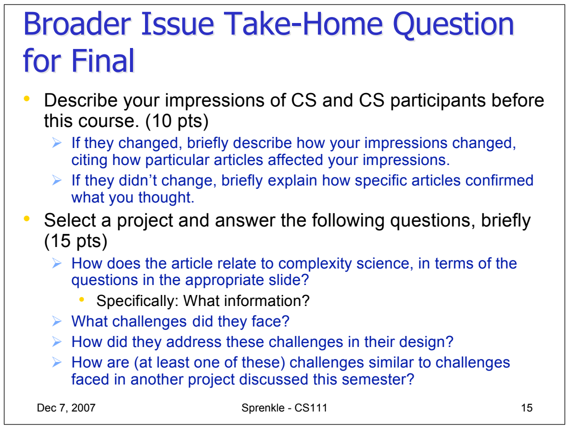
CS == Complexity Science

- Study of Complexity
  - How can it be done?
    - Based on **information**
    - Managing, manipulating data
    - Possible algorithms
  - How well can it be done?
    - Most **efficient** algorithm in terms of time and/or space
  - Can it be done at all?
    - Often, proof is a program--an implementation of the above



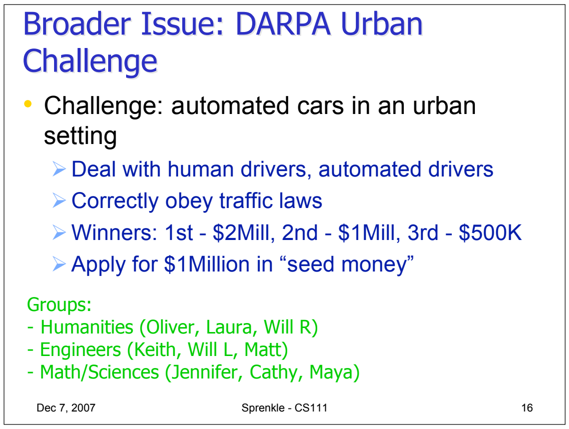
Broader Issues

- We've discussed different articles/projects throughout the semester
- Goal: you see how computer science and this course specifically relates to the world around you
- Interest score statistics:
  - Guesses on most interesting article?
  - Toughest "grader"?



Broader Issue Take-Home Question for Final

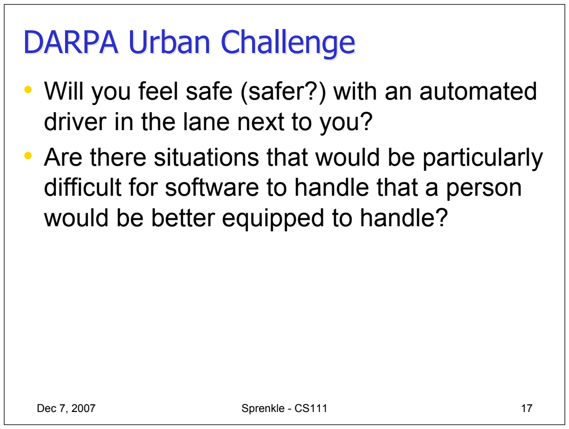
- Describe your impressions of CS and CS participants before this course. (10 pts)
  - If they changed, briefly describe how your impressions changed, citing how particular articles affected your impressions.
  - If they didn't change, briefly explain how specific articles confirmed what you thought.
- Select a project and answer the following questions, briefly (15 pts)
  - How does the article relate to complexity science, in terms of the questions in the appropriate slide?
    - Specifically: What information?
  - What challenges did they face?
  - How did they address these challenges in their design?
  - How are (at least one of these) challenges similar to challenges faced in another project discussed this semester?



Broader Issue: DARPA Urban Challenge

- Challenge: automated cars in an urban setting
  - Deal with human drivers, automated drivers
  - Correctly obey traffic laws
  - Winners: 1st - $2Mill, 2nd - $1Mill, 3rd - $500K
  - Apply for $1Million in "seed money"

Groups:
- Humanities (Oliver, Laura, Will R)
- Engineers (Keith, Will L, Matt)
- Math/Sciences (Jennifer, Cathy, Maya)



DARPA Urban Challenge

- Will you feel safe (safer?) with an automated driver in the lane next to you?
- Are there situations that would be particularly difficult for software to handle that a person would be better equipped to handle?