Computer Science I: Fundamentals (22C:016)

WELCOME!

INSTRUCTOR: SRIRAM PEMMARAJU

CONGRATULATIONS!

- This is an exciting time to be a computer scientist

- **Computational Thinking** is becoming part of all aspects of life:
  - Biology, medicine, biomedical engineering
  - Physics, astronomy
  - Economics, sociology
  - Music, Film
  - Humanities
  - …

Core CS Areas Are Thriving!

- Data mining
- Graphics
- Human Computer Interaction
- Networks
- Natural Language Processing
- Vision
- Algorithms
- Artificial Intelligence
- Operating Systems

Here is ACM's poster on careers in computing…
http://www.acm.org/membership/careernews/extras/careercolor.pdf

Watson vs The Humans!

- Mark your calendars: Feb 14-16
- A Jeopardy playing machine built by IBM Research will play human champions Ken Jennings and Brad Rutter
- Read about it in Wired magazine:
  http://www.wired.com/epicenter/2011/01/ibm-watson-jeopardy/

Microsoft's Kinect

- A controller-free gaming add-on to Xbox 360
- The Kinect sensor does full-body 3D motion capture, facial recognition and voice recognition.
- The software does motion analysis with feature extraction of 20 joints per player.

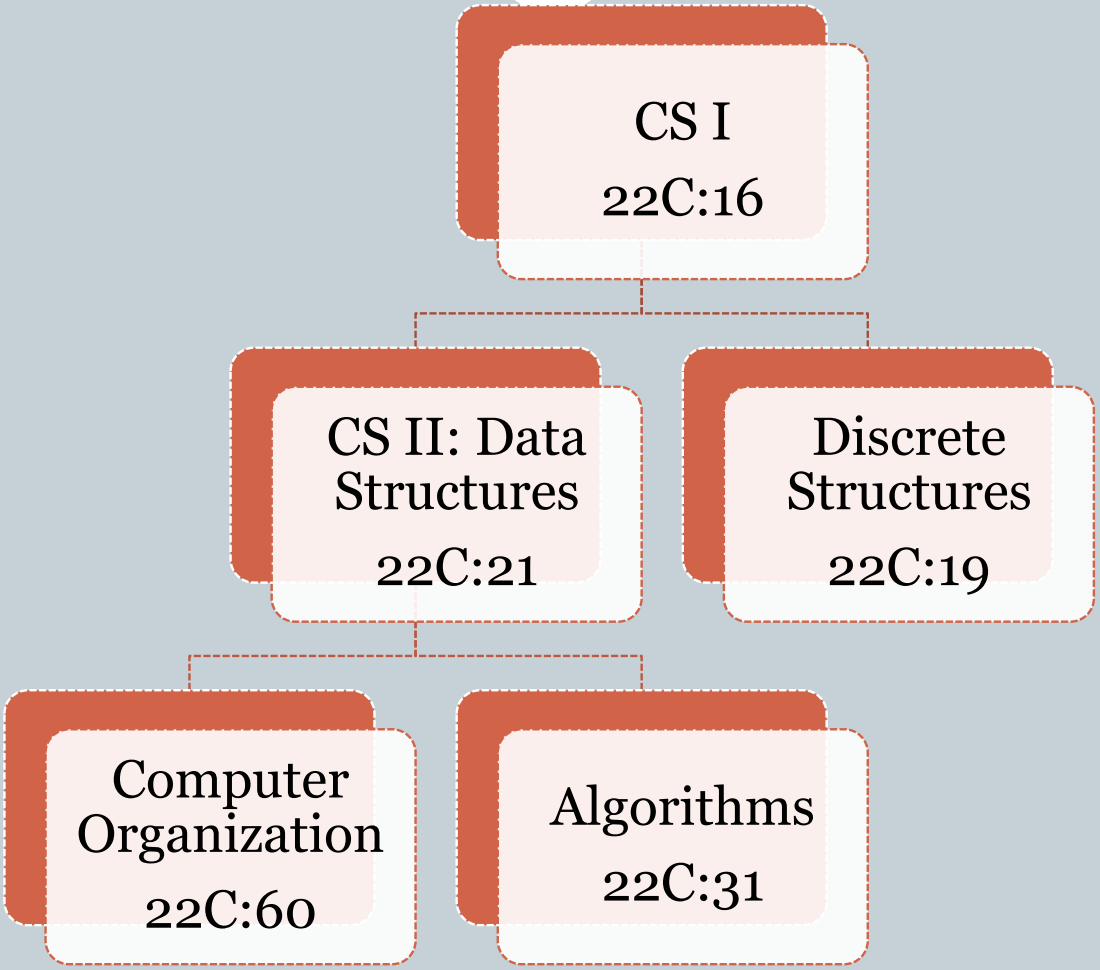
Computational Epidemiology at Iowa

- Computational tools to model, simulate, visualize and understand the spread of disease.
- Goal is to provide information to general public, hospital policy makers, etc.
- We use algorithms and graph theory, data mining, sensor networks, statistics, visualization,…
- Visit http://compepi.cs.uiowa.edu/

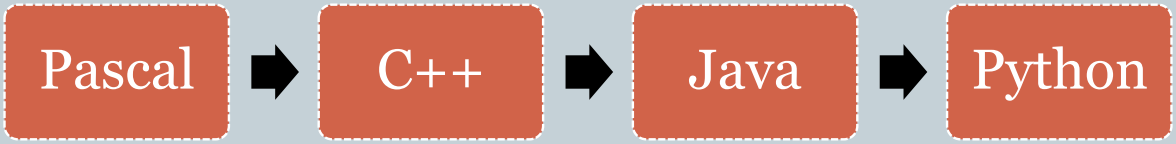
Computer Science I: Fundamentals

- Is much more than programming…
- A successful student will learn to view the world through a "computational lens."
- Introduction to
  - Designing algorithms
  - Thinking about their efficiency
  - Translating algorithms into reusable, reliable software

A Glimpse of what is ahead…



Programming Language: Python



Language in CSI over the years

A pitch for Python

- Easy to get started
- Allows beginners to focus on getting the computer to do what they want!
- Interactive mode is great for experimenting
- Extensive standard and third-party libraries
- No variable declarations, run-time rather than compile-time errors

To be successful…

- This should be the only course you are taking this semester!
- Separate **algorithm design** and **coding**
- Stay **unplugged** as much as possible
- Program **incrementally**, in tiny increments. And **test, test, test**…

See the Syllabus…

- For components that determine your grade
  - quizzes,
  - homework assignments,
  - programming projects, and
  - exams.
- And also for a note on expected amount of effort.

Help is plentiful

- Sriram Pemmaraju

  Coordinates
  101 G McLean Hall, 319 353 2956
  sriram-pemmaraju@uiowa.edu
  web: http://www.cs.uiowa.edu/~sriram/16/spring11

  Office Hours
  Monday 10:30-11:30
  Wednesday: 2:30-3:30
  Friday: 2:30-3:30

  You can make appointments or even just walk in

Teaching Assistants

- 3 CS PhD students will lead discussion sections (all on Tuesdays)

  Valerie Galluzzi (comp epidemiology)

  3:30-4:20 105 MLH, 4:30-5:20 105 MLH

  Thomas Hornbeck (comp epidemiology)

  12:30-1:20 105 MLH, 1:30-2:20 105 MLH

  Viet Thuc Ha (text mining)

  10:30-11:20 116 MH, 3:30-4:20 110 MLH

Students with disabilities

I would like to hear from anyone who has a disability which may require seating modifications or testing accommodations or accommodations of other class requirements, so that appropriate arrangements may be made.

Please see me right away.

Onto an unpleasant matter…

- There is no excuse for cheating.
- You cannot pass off someone else's work as your own.
- You can talk, but no actual exchange of written material.
- If you are not sure, see me right away.

The First Programming Problem

Write a Python program that reads a given positive integer and prints out the **binary equivalent** of that integer.

Example:

Input: 123 **Output:** 1111011

Input: 1363 **Output**: 10101010011

Input: 12 **Output**: 1100

Plan of action

1. Understand the problem. What does “binary equivalent” mean?
2. Design algorithms for the problem. How would we solve the problem with a pencil and paper?
3. Write down pseudocode for the algorithm.
4. Translate the pseudocode to Python code.
5. Test, test, test…

This example will illustrate…

- Constants
- Variables
- Operators
- Data types
- Expressions
- Function calls
- Input statements
- Output statements
- Program flow