COMP 10
EXPLORING COMPUTER SCIENCE
Lecture 1
Introduction

COURSE MECHANICS
Number: COMP0010B, Summer 2010
Meetings: Tuesdays and Thursdays, 9:00am-12:30pm
Lectures in Halligan 106
Lab exercises in Halligan 116
Instructor: Sara Su
sarasu@cs.tufts.edu
Halligan Extension 011

COMPUTERS ARE EVERYWHERE
How many computers have you used today?
How many are in this room?

AREAS OF COMPUTER SCIENCE
Here at Tufts, research areas include:
- machine learning
- computer vision
- programming languages
- algorithms
- robotics
- operating systems
- theory of computation
- computer graphics
- computational biology
- computer architecture
- human-computer interaction
- computational geometry
Theoretical concerns, as well as systems and applications

GOALS OF THIS COURSE
To gain an introduction to the field of computer science
To develop problem solving skills and computational thinking
To explore problem solving and programming techniques using the C++ language

TOPICS FOR TODAY
What is computer science?
What is a computer?
Computational problem solving
Computer programming
Introduction to Linux
WHAT IS COMPUTER SCIENCE NOT?

Not just about building computers or writing code!
These are the tools we use for computation

Learning to program is a good way to learn and practice the
process of identifying problems and designing and
implementing solutions.

COMPUTATIONAL PROBLEM SOLVING

Identify the problem
Design an elegant solution
Implement (automate) that solution

Discover and use the guidelines of how to compute, organize,
and process information

Computation: need for consistency and repeatability

COMPUTER SCIENCE

Study and applications
of the notion of Algorithm

ALGORITHM

“A precise step-by-step plan for a computational
procedure that begins with an input value and yields an
output value in a finite number of steps.” (source: Wikipedia)

COMPUTER

A general purpose machine
for manipulating data

COMPUTERS ARE DETERMINISTIC

Execute a set sequence of commands
Guided by current state and inputs
Always generate the same answer,
given the same inputs
A MODERN COMPUTER

INPUT DEVICES

Send information to a computer

OUTPUT DEVICES

Get information from a computer

CENTRAL PROCESSING UNIT (CPU)

Control Unit
Retrieves and decodes program instructions
Coordinates activities of all other parts of computer

Arithmetic & Logic Unit
Hardware optimized for high-speed numeric calculation
Hardware designed for true/false, yes/no decisions

MAIN MEMORY

Volatile - erased when program ends or computer turned off
Also called Random Access Memory (RAM)

Organized as follows:
- **bit**: smallest piece of memory
  - Has values 0 (off, false) or 1 (on, true)
- **byte**: 8 consecutive bits - bytes have addresses
- **words**: size depends on architecture (e.g., 32 bits or 64 bits)
- **subblocks**
- **blocks**

SECONDARY STORAGE

Non-volatile - data retained when program is not running or computer is turned off

Comes in a variety of media:
- **magnetic**: hard drive
- **Zip disk, floppy disk, tape**
- **optical**: CD-ROM, DVD, Blu-ray
CAPABILITIES OF THE MODERN COMPUTER

Precision, Speed, Size

Better than humans at:
  Sorting
  Searching, Retrieving
  Processing large amounts of data

Not as good as humans for other things...

SUMMARY: THE MODERN COMPUTER

Incredibly small, fast, and efficient

Deterministic: always responds the same way given the same commands

Capable of a variety of input and output

Connected to the world

Nothing until a programmer tells it what to be!

QUESTIONS?

PROBLEMS

A wide variety of problems can be solved with computers:
  Mathematical calculations, simulations, graphics, data analysis, robotics control, trend prediction, scheduling, word processing, presentations

These problems are rarely expressed as sequences of bits.

WHAT IS PROGRAMMING?

Programmers stand between the world of problems and the world of the computer

Take a problem
  Analyze the problem
  Envision a way to solve it
Take a computer
  Encode the solution
  Test & “debug”
  Maintain the solution

PROGRAMS

A program is a set of instructions to a computer

Computers are general-purpose machines
  Just about useless without a program

Programs turn computers into special-purpose devices capable of solving specific problems!
PROGRAMMING SKILLS

A mix of high-level creativity and low-level details

Programmers must learn to
- Be creative problem solvers
- Be meticulous artisans
- Think about computers at many levels
- Be patient and persistent!

MACHINE VS. HIGH LEVEL LANGUAGES

Computer hardware carries out instructions written in a **machine language**
- 1's and 0's – hard to understand or write!

A **high level language** is a notation that humans can understand and use more easily

A **compiler** translates a high level language to a machine language that can be executed

EVOLUTION OF LANGUAGES

Languages change as needs and technologies change

- **Machine Language** - 1940's
- **Fortran, Lisp** - 1950's
- **Cobol, Algol, APL, PL/I** - 1960's
- **Basic, Pascal, C** - 1970's
- **Smalltalk, C++, Modula, Ada, Prolog** - 1980's
- **Java, Php, Python, Ruby** - 1990's

SUMMARY: WHY STUDY COMPUTER SCIENCE?

Computers are everywhere, gaining ground, getting smaller and faster every year

Many of the problems that need solving today can be solved (or assisted) by computers
- Especially problems that are difficult for humans

The skills of programming enable us to create new solutions and better use existing ones

COURSE ROADMAP

Translate English into an **algorithm** and then into a **program**
- Which algorithm is more efficient?
- Will all programs eventually terminate?
- Cool programming assignments – simple to complex

Key concepts:
- The tools of programming
- The rules of programming
- The way of the programmer
TOOLS OF PROGRAMMING
Techniques, ideas, terminology, and constructs that make programming possible and effective

Problem analysis
Divide-and-conquer
Conditionals, iteration, recursion, etc.
Test-case design

RULES OF PROGRAMMING
We will use the C++ programming language
Widely used and industrial strength

The constraints and syntax of C++ and computers in general
Expressing programming constructs
Understanding and debugging errors
Familiarity with “artificial” languages

THE WAY OF THE PROGRAMMER
Style and patterns of thought that make programs successful and comprehensible

Not “necessary” parts of the program
Not required by the language
Key to becoming a truly proficient programmer
Key to creating elegant, beautiful code

QUESTIONS?

COMP 10
Prerequisites: No prior programming experience expected!

Fundamental concepts and algorithms
Programming assignments (in class) using C++
Final project on in-depth topic of your choosing

COURSE MECHANICS
Number: COMP0010B, Summer 2010
Meetings: Tuesdays and Thursdays, 9:00am-12:30pm
Lectures in Halligan 106
Lab exercises in Halligan 116
RECOMMENDED READING

Available at Tufts bookstore, online, and as an E-book
Also check Tisch Library course reserves

EVALUATION

Approximately:
- 70% Assignments
- 10% Final project
- 10% Quizzes
- 10% Class participation

EXPECTATIONS

- Ask questions
- Help one another, but do not share code
- Get ideas from books/online
- Always cite sources
- Challenge yourself

ACCOMMODATIONS

To request an accommodation for a documented disability, you must register with the Disability Services Office at the beginning of the semester

Call (617) 627-2000 to arrange an appointment with Sandra Baer, Program Director of Disability Services

BOOKMARK THIS!

http://www.cs.tufts.edu/comp/10EXP

THE NEXT SIX WEEKS

Some tools
Some rules
The way is up to you...
**LINUX**

The **operating system** we will use in this class

Free, open source software

All source code can be freely modified and redistributed under free software licenses

**OPERATING SYSTEM**

Program that controls hardware, runs other programs, handles input/output, manages other devices

e.g., Windows, Mac OS, Linux, Unix

**LINUX OS**

- **Kernel** (around since 1991)
  - **shell**
  - **desktop environment**
  - **utilities** (applications)
**LINUX OS**

- Used on
  - Servers
  - Desktop/laptop computers (Halligan 116 & 118)
  - Cell phones
  - Supercomputers

- “Flavors” of Linux (a.k.a. distributions) include
  - Ubuntu
  - Fedora (Red Hat)

**USEFUL FOR…**

- Text editing: **emacs**, vi
- Web browsing: Internet Explorer, Firefox
- Email: WebMail, pine
- Programming: **g++**, Eclipse
- Math, graphics, gaming
- Office applications
- Manipulating the file system

**SHELL**

Intermediary program between user and OS
- Reads commands, sends to OS to be executed
- Lets us manipulate **file system** and run programs

**FILE SYSTEM**

- Consists of
  - **Files** that contain data
  - **Directories** that contain files

- Directories are organized in a tree structure

- root node
- leaf nodes
TREES AND PATHS

-username/CS/Hello.cpp
-username/CS/

~username/CS/Hello.cpp
~username/CS/

refers to a file
refers to a directory

SOME COMMON LINUX COMMANDS

Getting help:
man
Navigating the file system:
ls, cd, pwd
Manipulating the file system:
rm, cp, mv, mkdir, rmdir
Viewing files:
cat, more, head

EMACS

A widely used text editor
Similar in some ways to Notepad

Will get familiar with editing files in Emacs in lab

PREVIEW: FIRST LAB

Log in to a computer with your CS username & password
Get familiar with the Linux system
Start using the Emacs text editor
Start entering commands

QUESTIONS?

HOT TOPICS IN COMPUTER SCIENCE
COMPUTATIONAL BIOLOGY

BEYOND TRADITIONAL ELECTRONICS

DNA computing
Quantum computing
Etc.

HUMAN-COMPUTER INTERACTION

ROBOTICS & ARTIFICIAL INTELLIGENCE

AUTONOMOUS ROBOTS

OR NOT?
TO DO

Before next class…

Visit the class webpage:  http://www.cs.tufts.edu/comp/10EXP
Create a CS account:  http://www.eecs.tufts.edu/~accounts
Recommended reading:  Horstmann, pp. 21-26, 34-44, 50-53