Welcome to COMP 40!

Noah Mendelsohn
Tufts University
Email: noah@cs.tufts.edu
Web: http://www.cs.tufts.edu/~noah
Course Overview
COMP 40 Goals

- Learn how computers work
- Turn the corner toward being truly *professional* programmers
How we’ll do it

Ramp up your Programming Skills

Big programs that teach you abstraction, pointers, locality, machine representations of data

Intel Assembler Programming
*The Bomb!*

Building Useful Applications in your Language

Building a Language Processor on your Emulator

Emulating your own hardware in software
How we’ll do it

Ramp up your Programming Skills

Big programs that teach you abstraction, pointers, locality, machine representations of data

Building Useful Applications in your Language

First assignment:
1) Programming skills
2) Hanson interfaces: practice in abstraction
3) Models of how abstract types are implemented
4) void * practice

Intel Assembler Programming

The Bomb!
How we’ll do it

Ramp up your Programming Skills

Big programs that teach you abstraction, pointers, locality, machine representations of data

Next three assignments:
1) How data is represented
2) How languages and libraries use the machine
3) Locality

Intel Assembler Programming

The Bomb!

in software
How we’ll do it

Ramp up your Programming Skills

Big programs that teach you abstraction, pointers, locality, machine representations of data

Building Useful Applications in your Language

Building a Language Processor on your Emulator

Assembler programming on popular machine in software

Intel Assembler Programming: The Bomb!
How we’ll do it

- Ramp up your Programming Skills
- Building Useful Applications in your Language
- How machines run internally
  - Machine representations of data
  - Intel Assembler Programming *The Bomb!*
  - Emulating your own hardware in software
- Building a Language Processor on your Emulator
How we’ll do it

Ramp up your Programming Skills

Build your own versions of the tools you’ve been using

Building Useful Applications in your Language

Intel Assembler Programming
The Bomb!

Emulating your own hardware in software

Emulating your own hardware in software

Intel Assembler Programming
The Bomb!
How we’ll do it

Ramp up your Programming Skills

Big programs that teach you abstraction, pointers, locality, machine

Building Useful Applications in your language

Building a Language Processor on your Emulator

The Bomb!

Use what you built!

Use what you built!
How we’ll do it

- Ramp up your Programming Skills
- Building Useful Applications in your Language
- Emulating your own hardware in software
- Intel Assembler Programming
- The Bomb!

ABSTRACTION!
How we’ll do it

Ramp up your Programming Skills

Big programs that teach you abstraction, pointers, locality, machine representations of data

Intel Assembler Programming The Bomb!

Building a Language Processor on your Emulator

Building Useful Applications in your Language

Emulating your own hardware in software
Important topics for the term

- Abstraction
- How machines work
- Data representation – what is a bit?
- How languages and runtimes use the machine
- The skills and mindset of a great programmer
Becoming a great programmer

- **Design** thoughtfully before you code
  - Writing down your design helps you think clearly about it and share it with others

- **Write for people:** Work read by people as well as machines
  - Your code should be beautiful and easy to understand
  - Prose (English) documentation *is as important as your code!*

- **Testing** is as important as design and coding

- **Techniques** for successful programming. You need
  - Patience, perseverance, good rest 😊, humility, confidence
  - Insight into good programming & testing techniques
  - Often: teamwork, etc., etc.

THESE ARE THINGS THAT TOP PROFESSIONAL PROGRAMMERS DO!
Key things to watch for throughout the term

- Design thoughtfully before you code
  - Writing down your design helps you think clearly about it and share it with others
- Your work is read by people as well as machines
  - Your code should be beautiful and easy to understand
  - Prose (English) documentation is as important as your code!
- Testing is as important as design and coding
- Successful programming requires
  - Patience, perseverance, good rest
  - Humility, confidence
  - Insight into good programming & testing techniques
  - Etc., etc.

HELPING YOU DEVELOP THESE SKILLS IS A KEY GOAL OF COMP 40

THESE ARE THINGS THAT TOP PROFESSIONAL PROGRAMMERS DO!
Credits

- This version of COMP 40 was originally developed by Prof. Norman Ramsey
- Many of the materials and assignments trace to him
- Updates have been made by Noah Daniels, Mark Sheldon, and Noah Mendelsohn
Logistics
Course staff

- **Professor:**
  - Noah Mendelsohn (please call me “Noah”)
  - Emails: noah@cs.tufts.edu
  - Office hours: Tues/Thurs 3:15PM – 4:15PM (after class)

- **Graduate TAs**
  - Noah Cooper, Patrick Kinsella, Sam Hincks, Behnam Heydarshahi
  - Email: comp40-staff@cs.tufts.edu

- **Undergraduate Tas**
  - TA Fellows: Chris Phifer and Tom Hebb
  - Too many to list: all COMP 40 experts – *times on Piazza; find them in the labs!*
  - Email: ta40@cs.tufts.edu
What you must do right now!

- Make sure you have a partner for assignment 1
- Read course: [home page](#), [admin & policies](#) including collaboration policy, pair programming rules, coding standards
- Sign up for [Piazza](#):
  - We mostly won’t email announcements – just post there
- Start learning C and C++ (see [self-study module](#) with required submission before Friday labs)
- Immediately: start reading the [HW1 handout](#) and readings from [course calendar](#) especially those from Hanson’s book
- Look for other tutorial material linked from calendar/homework
- Come to lab on Friday with your partner
That’s a lot to juggle at once!

- We know…
- Your main goal for the next two weeks is to master HW1
- The self-study material and labs are there to help you learn
- HW1 and the HW1 design document are far more significant to your grade than the self-study material or labs…but lab attendance is *required*
Homework 1 is out now

- Also linked from course calendar
- Due dates:
  - Design document: Submit by **Monday September 11**
  - Final submission with code: **Friday September 15**
- Be sure to understand course policy on late submissions: up to **2 tokens/assignment, 6 total for term. No credit if > 2 days late!**
- See admin page for policies on serious illness, etc.
Pair programming

- You must work *with* a partner – rare exceptions
- Both of you **must be together** when design, documentation, or coding is done!
- You must **never** split the work
- After the first assignment you choose your own partner – max 3 assignments with same partner
- Be sure to understand and follow all pair programming rules at: [http://www.cs.tufts.edu/comp/40/admin#pairprogramming](http://www.cs.tufts.edu/comp/40/admin#pairprogramming)
- We will discuss in detail Thursday
Labs

- Required – attend with your partner
- We will, by next week, activate the online system to allow you to switch labs on a space-available basis
- Many labs are designed to give you essential help completing your big projects!
A few more thoughts

- This course is intense
- Most students report it is very worthwhile
- Grading can be severe on individual assignments but…
- …most students do very well in the end
- *We are here to help you succeed!*
- Relax…work hard…be patient…have fun!
Design Documents
Why write down your designs?

- Forces you to think clearly and make deliberate choices
- Changing a written design is (usually) easier than changing code
  - Your first design will probably not be a good one!
  - Can have debates at higher, algorithmic level without distraction of code
  - Keep refining until the design feels right
- Your code is more useful if other people can understand it: design becomes an important artifact for documenting work
- On real programming projects, designs are debated before code is written!
COMP 40 Design Documents

- 2 kinds: for programs and for ADTs (We’ll discuss ADTs later)
- Three layers documented in separate sections
  - Architecture: the big picture
  - Interfaces: how pieces communicate…what they hide from each other
  - Implementation: how the pieces are built
- Invariants
  - Important concept in computing
  - Representation/structural invariants & loop invariants
  - We’ll discuss invariants in the next lecture
- COMP 40: special documentation for ADTs (to be discussed)

Important information about writing Design Documents can be found from the “References” tab of the COMP 40 Website: http://www.cs.tufts.edu/comp/40/reference/
COMP 40 Design Documents

- Three layers documented in separate sections
  - Architecture: the big picture
  - Interfaces: how pieces communicate…what they hide from each other!
  - Implementation: how the pieces are built

- Invariants
  - Important concept in computing
  - Structural invariants & loop invariants
  - We'll discuss invariants in the next lecture

- COMP 40: special documentation for ADTs (to be discussed)

Important information about writing Design Documents can be found from the “References” tab of the COMP 40 Website:
http://www.cs.tufts.edu/comp/40/reference/

For HW#1 (filesnpix)
You will submit a simplified design document that just explains your data structures and testing plan.

See the handout for information about what’s required.
Getting Help
Where to get help

- **Halligan is the place to get help with your code!**
  - TAs: in labs most days and evenings
  - I’m around a lot: see office hours on my home page
  - Schedule will be kept on Piazza page

- **Piazza!**
  - *We much* prefer you ask questions there
  - *You must not* share your own code or test cases with other students: mark those questions “instructor only”
  - Don’t post anonymously: we’re a mutually supportive community…but it’s your choice

- **Online sources**
  - man pages, Google, Stack Overflow
## IMPORTANT: Collaboration rules

<table>
<thead>
<tr>
<th>Who</th>
<th>Allowed</th>
<th>Not Allowed!</th>
</tr>
</thead>
</table>
| Current COMP 40 students other than your partner | • Teaching each other technology (C, caches, etc.)  
• Discussing overall approaches to HW problems  
• Debugging compile scripts, sharing editor tricks, etc. | • Showing others your solution code  
• Sharing or looking at other people’s solution code  
• Sharing test cases |
| Friends, family, former COMP 40 students, etc. | • Studying technology (C, caches, etc.)  
• Debugging compile scripts, sharing editor tricks, etc. | • Technical discussion of homework assignments  
• Having others look at or help debug your code or test cases  
• Having others show or give you their solution code |
| COMP 40 TAs and Staff | • May help you with all aspects of your work  
• Rarely: may authorize others to help with your code | • Staff will help you learn to find answers, but will generally avoid giving away answers |

You must *never* seek out or look at solutions to current or past COMP 40 assignments!!
Observing the rules

- When getting help from others you **must** observe rules from the previous slide!

- **Yes: many of our assignments are similar previous terms:**
  - Why: inventing assignments this good takes several months each
  - *You* are responsible for avoiding the temptation to seek out solutions from friends or on the Web
  - You must not show your code to anyone other than partners and course staff
  - The consequences for violating Tufts rules on academic integrity can include suspension from the University, etc. DON’T DO IT!
  - Remember: the goal is to learn, not to get the assignments submitted!

- **If you slip up and break the rules…tell us immediately and in detail!**
  - The penalties will often be much less severe if you admit honestly and promptly to minor mistakes

**REMEMBER:** Halligan is the place to get help with your code!
“If you give a man a fish he is hungry again in an hour. If you teach him to catch a fish you do him a good turn”

Anne Isabella Thackeray Ritchie
In COMP 40, we will sometimes decline to give you fish, but we will always help you learn to fish.
But...just this once since we’re hitting you with a lot at once...

...I’ll give you a few fish!