Welcome!

COMS 3157

Advanced Programming

Fall 2017

Teaching staff

- 15 Teaching Assistants (TAs), all former 3157 students
  - Kevin Chen <kxc2103@columbia.edu> - Head TA
  - Emma Etherington <ele2116@columbia.edu> - Head TA
  - Joshua Zweig <jmz2135@columbia.edu> - Head TA
  - John Hui <jzh2106@columbia.edu>
  - JiaYan Hu <jh3541@columbia.edu>
  - Katie Stein <kls2210@columbia.edu>
  - Nelson Gomez <ng2573@columbia.edu>
  - Evan Wickenden <eww2119@columbia.edu>
  - Ivy Chen <ic2389@columbia.edu>
  - Emily Jin <ej2332@columbia.edu>
  - Jennifer Wenjun Bi <jb3495@columbia.edu>
  - Elshadai Tesfaye Biru <etb2119@columbia.edu>
  - Dean Deng <dd2563@columbia.edu>
  - Aunoy Poddar <ap3441@columbia.edu>
  - Tony Qian <tq2115@columbia.edu>

Teaching staff contact info

- TA email & office hours
  - Email to cucs3157-tas@googlegroups.com goes to all teaching staff
  - TA room – 1st floor, Mudd building
  - TA calendar: http://bit.ly/3157-cal (will be filled by this weekend)
- Instructor email & office hours
  - Jae Woo Lee jae@cs.columbia.edu – 715 CEPSR
  - Jae's calendar: http://bit.ly/jae-cal (this week's OH posted; rest will be filled by this weekend)

Who am I?

- Jae Woo Lee
  - Senior Lecturer in Computer Science
    - Teaching first, research second
  - Just call me Jae (pronounced 'Jay')
    - Note that this is NOT a general rule – address instructors as Professors unless told otherwise
- My background
  - Undergrad in Columbia College
  - Many years of professional experience
    - Designing and coding large-scale software systems
    - Running a start-up company
  - Came back to Columbia for Ph.D.
  - More info at http://www.cs.columbia.edu/~jae/

Reviews

"Jae is a fantastic lecturer."
"Jae Lee is a terrible professor. I wouldn't even want him as a TA for this class."
"The best! His remarks will live with me for the rest of my career."
"Jae Lee is the worst human being I have ever had as a professor."

[. . .]

"You will learn a lot. Just ignore Jae."

Sources:
CULPA - http://culpa.info/professors/3509
Spring 2017 Course evaluations - http://www.cs.columbia.edu/~jae/evals/3157-eval-final-2017-1.pdf

This course

According to BWOG:

One of "The Best Classes Ever"

One of "Classes To Take Before You Die I Mean Graduate"

- Introduction to systems programming
  - Arguably the most important course in CS curriculum
- *Follow the River and You Will Find the C*
  - Paper published in SIGCSE 2011
  - Describes this course: what, how, and why
  - Great overview of what you are in for

(See `http://www.cs.columbia.edu/~jae/` for links)

Registration

- Two sections
  - See class home page for details: http://www.cs.columbia.edu/~jae/3157/?asof=20170904
  - Lectures will be identical
  - Personal recording allowed
    - Share with class, but not on the Internet
- R credit not allowed
- Auditors are welcome to lectures
  - But no Canvas; no Linux account; no homework; no exams; no recitations
- SPS students must contact Dean's office
  - Registrar told me never to sign add-drop

Review sessions

- Logistics
  - One topic / week, multiple sessions by different TAs
  - Most likely evenings between Friday and Monday
    - Time and place TBA
  - Attendance optional, but recommended
- Topics
  - UNIX basics, editors, Git, etc. (in the beginning)
  - Lecture reviews
  - Lab assignment clarifications & reviews
  - Lab solutions walk-through
  - Exam preps

Participate in class, please!

- Classes is no fun (for me, at least) if we don’t interact
  - Answer questions I pose
  - Ask questions anytime
  - Embarrass me when I’m wrong
- People are afraid to ask when they think:
  - “I’m the only one who doesn’t know this.”
  - “I can’t frame this question clearly and eloquently.”
  - “Maybe he just said it when I dozed off just now…”
- Big class, so I may not entertain all questions, but:

NEVER BE AFRAID TO ASK ANYTHING, IN THIS CLASS AND IN LIFE!

Prerequisites

- Absolutely required:
  - 2 or 3 semesters of Columbia-level programming courses
    - Ex) 1006-1004-3134; 1007-3137; etc.
- Pretty much required
  - Data Structures (3134 or 3137)
    - For general CS & programming maturity
    - Ex) I'll assume you know all about recursion
    - DON'T take DS and 3157 together – too much programming at once
- Recommended:
  - Familiarity with UNIX environment – if not, learn ASAP
  - Knowledge of Java – only to draw comparisons with C++
- No C/C++ knowledge assumed

Course objective

- Simply put:
  - Right now, you are a programming student
  - After this course, you will become a *programmer*
- How?
  - Move beyond Java
    - Learn C/C++
    - Become proficient in UNIX programming tools
  - Move beyond toy programming
    - Learn advanced techniques used in real-world software
    - Learn design principles used in large-scale software

Why C?

- It's cool
  - There are two kinds of programmers: those who know C and those who don't
    - *Corollary*: There are two kinds of *Java* programmers: those who know C and those who don't
  - Your kung fu will be better than theirs
- It's fundamental
  - Understand how computers work
- It's useful
  - Get ready for OS class
  - Build foundation to learn C++ and Objective-C

Topics covered

Course is divided into 3 parts:

1) C
   - Mastery of C language is the most important part
   - Everything else depends on it!
2) UNIX systems programming
   - Process control, signal, I/O, TCP/IP networking
   - Sockets API and HTTP protocol
     - Write your own web server from scratch!
3) C++
   - C++ language: we will not cover everything
   - Generic programming: templates and STL

COMS 3136 for non-CS majors

- COMS W3136 Essential Data Structures in C/C++
  - Please consider 3136 if you're not a CS major
  - Usually offered in Fall semesters
  - Fall 2017: TR 5:40pm-6:55pm, 833 Mudd
- A fusion of 3157 and 3134
  - 3157-lite: C & C++, but no heavy systems stuff
  - 3134-extract: only the most important data structures
  - Bridges E1006 and many 4000-level CS courses
  - Perfect for EE & IEOR folks who came to 3157 to learn C/C++ but found it a bit too much

Grading

- Grading logistics may change later
- You get overall score out of 100, comprised of:
  - Lab assignments – 25%
  - Midterm exam #1 – 20%
  - Midterm exam #2 – 25%
  - Final exam – 30%
- I look at everyone's lab & exam scores in a big spreadsheet sorted by the overall score
- I decide cutoffs for letter grades A+, …, D, F
  - No predetermined formula
- Booster: I reserve the right to raise one's overall score by a small amount (typically less than 0.5%)

Booster

- Grade boost based on subjective evaluation
  - Most people will not get it
  - Have been used to boost some borderline cases
  - Can be up to 5% in theory, but never been > 1%
- Based on:
  - Class participation
  - Mailing list participation
  - Beautiful code
  - Awesome documentation
  - Optional work

10 assignments (aka labs)

- Some labs may not be graded
  - A random subset of at least 7 out of 10 will be graded
    - Assume that a lab is graded unless I say otherwise after the deadline
  - Lowest score will be dropped (i.e. converted to zero)
    - In other words, everyone is forced to get zero on one lab
    - Note that labs have different weights (between 100 and 150), so you'd be at a disadvantage if you end up dropping a bigger one
- Deadline
  - Soft deadline, and then hard deadline 2 days later
    - You use 1 late day if you submit within 24 hours after the soft deadline
    - You use 2 late days if you submit between 24 and 48 hours after the soft deadline
    - After 48 hours past the soft deadline, no submission will be accepted
  - You have 7 late days total; up to 2 can be used for a single lab
    - Check your late days by running: `/home/w3157/submit/check-late-days`
  - Absolutely no exception under any circumstances
  - After you receive grade, you have 2 weeks to send re-grade request

Lab grading

- Grading model
  - You are a software company
  - I hire you to develop a product according to spec
  - You ship the finished & polished product on time
  - TAs are the end users who will pay you with grade
- What this means:
  - Your software doesn't work, they don't pay
  - Your software didn't follow spec, they don't pay
  - Your software didn't ship on time, they don't pay
  - But you worked so hard… they sympathize, but they don't pay
- For example:
  - Your software doesn't compile – you get ZERO
  - Deductions for not following spec EXACTLY
    - Ex) Spec asked for README.txt file, not README, not README.md, not Readme.txt

Exams

"This class is amazing. Jae is amazing. The TAs are amazing. There's not much bad to say about it other than... Oh. Right. The tests."

- Anonymous, CULPA

How to do well on exams

Exams are normally closed-book, written, and based on labs and lectures.

So I suggest you should:

1. Do the labs. I mean, *really* do the labs.
   - Don't let TAs fix your problems – you need that pain
   - Don't just "get it working" – understand every detail
   - Don't code by trial & error – understand your errors
2. Learn to read code on paper
   - Read & understand every line of solution code
   - Read & understand every line of sample exams
   - Read & understand code from the textbook
   - Then try coding them yourself without looking
3. Attend lectures and pay attention

Zero tolerance on cheating

- **REQURED READING:** http://www.cs.columbia.edu/~jae/honesty.html
- You are cheating if you:
  - Take code from friends, or search for code on the Internet
  - Look at solutions that your friend has from previous semester
  - Upload any class materials (including your own code) to public repository (ex. GitHub) during or after this semester
- We can tell
  - We compare you submissions to CURRENT AND PREVIOUS submissions
  - You submit work history – minimum 5 commits required
  - Once you look at cheat code, you won't be able to come up with anything else
- Result of cheating
  - Case 1: You get caught
    - Academic penalty – anywhere between 1 letter grade down and F
    - Referral to the Office of Judicial Affairs
    - 36 cases in Spring 2016; 34 cases in Fall 2016
  - Case 2: You get away with it
    - You will keep cheating for the rest of your life – have a nice life.

Class ListServ

- Communication between all of us
  - Official announcements, lecture notes, lab assignments
  - Should be the 1st place to go for non-personal questions
- Do:
  - Ask & answer questions
  - Provide helpful tips and fun links for your classmates
  - Be considerate & friendly
- Don't:
  - Ask questions without first trying to solve it on your own
  - Post code or critical info that leads directly to solution
  - Be impatient & rude
- TAs and I respond to emails in this order:
  1. All pending questions on the listserv first
  2. All pending questions sent to cucs3157-tas@googlegroups.com
  3. Then individual emails
  4. NEVER send a same question separately to multiple people
     - You will get banned from ever sending an email if you get caught doing this.

Manage ListServ emails

- Learn to manage high volume – filter by tags in subject
  - [cs3157] – all emails from the class listserv will have this tag
  - [ANN] – important announcements from me or TAs
  - [LAB*n*] – information relevant on a particular lab
  - Examples:
    - [cs3157][ANN] Sample midterm
    - [cs3157][ANN][LAB7] Correction on lab7 instruction
    - [cs3157][LAB6] in case you're curious about fdopen()
- Setup Gmail filters
- Keep up diligently

- Yes, I know about Piazza. Thanks for your suggestion.

Textbooks

- Required
  1. *The C Programming Language* (2nd ed.) – aka K&R C
     - By Kernighan and Ritchie
     - Simply the best
  2. *A Tour of C++*
     - By Bjarne Stroustrup
  - Survey in Spring 2016: only 4% bought them at the local bookstore
  - So get them wherever you usually get your textbooks
- Highly recommended reference for UNIX programming
  - *Advanced Programming in the UNIX Environment* (3nd ed.)
    - By Stevens & Rago

HW0: 50 points total

- **Part A (20 points): due Tuesday 9/5, 11:59pm (tonight)**
  1. Subscribe to 3157 ListServ today
     - https://lists.cs.columbia.edu/mailman/listinfo/cs3157
     - In the textbox “Your name (optional)” put Your Full Name (UNI)
       - For example: Jae Woo Lee (jwl3)
     - **You must reply to the confirm email (which might be in your spam folder)**
     - Then receive “Welcome to the "Cs3157" mailing list”
       - This email contains your password for accessing archives of past postings
     - All emails to listserv, TAs, or me MUST include your UNI
       - Sign it with UNI if you don’t use UNI@columbia.edu
  2. Get the textbooks
     - Start reading K&R chapters 1,2,3,4

HW0 continued

- **Part B (30 points): due Thursday 9/7 11:59pm**
  1. Read the following two documents:
     - http://www.cs.columbia.edu/education/honesty
     - http://www.cs.columbia.edu/~jae/honesty.html
  2. Send me an email containing:
     - Subject: “[3157] hw0-UNI”
       – Without the quotes, sole space before hw0, UNI replaced with your actual UNI in lowercase
     - Your name, major & school program, year
       – Ex) Jae Woo Lee, Physics, Columbia College, class of 1994
     - Your pledge
       – see honesty.html above
     - CS classes taken and/or other programming background
     - Optionally anything else you want to let me know
     - Optionally attach a picture of you, but please reduce image file size to about 100KB