CS Systems Qualifying Exam 2019

Important Dates

- **May 2**: Registration ends. Registration instructions are below. When you register, you must declare the three exams you plan to take.
- **May 20**: Exams. Candidates will take exams during the week starting Monday, May 20. For oral exams, students are responsible for scheduling an exam with the examiner. We will schedule the written exams.
- **May 27**: Grading complete. Faculty members email grades for all exams to Subhasish Mitra **(and Cc Beverly Davis)**.
- **Tentatively May 31**: Faculty meet to decide on final grades **(Time TBD)**. All systems faculty are invited.
- **June 3**: Announce. Decisions emailed to students and advisors. You should only take the Systems Qualifying Exam if your adviser is a “Systems Faculty” member. Students without a Systems Faculty adviser may request to take the exam, but you need the permission of your advisor and Subhasish Mitra.

Exams

- Architecture: Kunle Olukotun
- Compilers: Monica Lam
- Databases: Peter Ballis
- Graphics: Doug James
- Networking: Sachin Katti
- Operating Systems and Distributed Systems: David Mazieres
- Security: Dan Boneh

Registration

Send an email to Subhasish Mitra (subh@stanford.edu) **and Cc Beverly Davis** (beverlyd@stanford.edu) indicating which of the three exams you wish to take.

Architecture Qualifying Exam

Details and Reading List

Examiner: Kunle Olukotun
This qual will cover any ideas, topics and reading covered in the following courses:

- EE282: Computer Systems Architecture
- EE382A: Advanced Processor Architecture
- CS315A: Parallel Computer Architecture and Programming

You will be expected to be very conversant with the key ideas in computer architecture: Levels of abstraction (e.g. ISA→processor→RTL blocks→gates), pipelining, caching, prediction, virtualization and parallelism. As someone taking the architecture qual, you will be expected to have a fairly sophisticated knowledge of these topics.

Reading: Computer Architecture: A Quantitative Approach, 3rd Edition , Hennessy & Patterson.

Format

30 minute oral exam.

Scheduling

Arrange with Kunle Olukotun <kunle@stanford.edu>

Compilers Qualifying Exam

Details and Reading List

Examiner: Monica Lam

Principles, Techniques, & Tools (Second Edition), Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, Addison Wesley, 2007.

Format

Oral or written exam, depending on the number of exam takers.

Scheduling

Contact Monica Lam <lam@cs.stanford.edu> directly.

Databases Qualifying Exam

Details and Reading List

Examiner: Peter Bailis

This qual will cover any ideas, topics and reading covered in the following courses:

- CS145: Introduction to Databases
- CS245: Database System Principles

You will be expected to be conversant with key ideas in data management, including: data models and query languages; database design; building database applications; file structures, indexing, and hashing methods; query processing and optimization; Transactions, concurrency control and recovery; security and authorization; and database performance. As someone taking the database qual, you will be expected to have a fairly sophisticated knowledge of these topics.

Reading: H. Garcia-Molina, J.D. Ullman, and J. Widom. Database Systems: The Complete Book (Second Edition) Prentice Hall, 2009.

Format

30 minute oral exam.

Scheduling

Contact Peter Ballis <pbailis@cs.stanford.edu>

Graphics Qualifying Exam

Details and Reading List

Examiner: Doug James

Knowledge of basic representations of surfaces and solids, scan conversion algorithms, geometric transformations, viewing projections, visible surface determination, and shading models, as covered in CS 148 and 248. Understanding of basic issues in input and display hardware, graphics software, and user interface design. Some knowledge in depth of either curve, surface, and solid modeling and geometric algorithms at the level of CS 348A, or of sampling, filtering, and local and global illumination methods at the level of CS 348B, or of computer animation and physics-based simulation at the level of CS 348C.

Reading list:

Typical references for the material in 248, 348A, 348B and 348C, are the online course notes and references, where available:

- For 148: P. Shirley,, Fundamentals of Computer Graphics, 2nd Edition, A. K. Peters, 2006
- For 248: E. Angel, Interactive Computer Graphics (5th ed.)
- For 348A: G. Farin, Curves and Surfaces for Computer Aided Geometric Design Academic Press, (2nd edition)
- For 348B: M. Pharr and G. Humphreys, Physically Based Rendering. Morgan Kaufman.

Format

30 minute oral exam.

Scheduling

Contact Doug James <djames@cs.stanford.edu>

Networking Qualifying Exam

Details and Reading List

Examiner: Sachin Katti

The reading list from CS244: http://web.stanford.edu/class/cs244/timetable.html

Format

The exam is an oral examination with Sachin Katti.

Scheduling

Contact Sachin Katti <skatti@stanford.edu>

Operating Systems and Distributed Systems Qualifying Exam

Details and Reading List

Examiner: David Mazieres

Reading List:

Alexander Matveev, Nir Shavit, Pascal Felber, and Patrick Marlier. Read-Log-Update. In Proceedings of the 25th ACM SOSP. 2015.

Format:

Implement and benchmark a user-level RLU library in C++14 (or later) using C++ atomics. You should support the following functions described in the paper: rlu_reader_lock, rlu_reader_unlock, rlu_dereference, rlu_try_lock, rlu_cmp_objs, rlu_assign_ptr, rlu_abort, rlu_malloc, and rlu_free. Note, however, that you don't need to have functions by these names or keep the exact same parameter lists, because you may be able to expose the same functionality implicitly using templates and operator overloading.

Email two things to David Mazières <mazieres-7iq6g48fqmkhjnxvzr9fihkb9i@ta.scs.stanford.edu> **by 11:59pm Friday May 24, 2019:**

1) A tarfile of your implementation, and

2) A write-up of no more than 3 pages with no smaller than 11pt font and 1-inch margins. Your write-up must contain at least one graph with benchmark results.

If you wish, you can assume that your compiler properly implements memory_order_consume, though this is now deprecated. Alternatively, you can include some non-portable code, so long as you explain why and keep the non-portable code to a minimum. You can borrow code from the linux kernel (or other open-source operating systems) for the non-portable parts. Your implementation only needs to work on one architecture with one compiler.

Security Qualifying Exam

Details and Reading List

Examiner: Dan Boneh

Format

http://seclab.stanford.edu/SecurityQual.html

Scheduling

http://seclab.stanford.edu/SecurityQual.html