THE CISE QUALIFYING EXAM 1 (QE 1): January 2012

Please note:

The scopes of the QE 1 may change from year to year. This page details the scopes for the exam to be given in January 2012.

Contents

Purpose of Qualifying Exam 1 (QE 1)

The QE 1 is designed to have Ph.D. students demonstrate the following abilities:

- Ability to think critically and reason about domain-specific situations
- Ability to incorporate new definitions/information/etc in domain-specific situations

Specifically, we expect students to demonstrate their ability to think critically about situations related to their field of study. We are NOT interested in students' ability to memorize specific topics. Therefore, we have identified narrow scopes for each of the six areas the QE 1 covers:

- Each of the six areas has one or two specific references listed.

  These references, along with the specific chapters/sections listed, represent the level of knowledge we expect students to have in a certain area. **Specifically:** We expect students to have sufficient knowledge and ability to be able to successfully solve all problems listed in that chapter/section.

  Caveat: The above statement **does not mean** that we will ask you questions that appear in that specific reference. It means that, *if you can successfully solve all the problems listed in that chapter/section*, then you should also be able to successfully complete the QE 1 questions.

- As you prepare for the QE 1, you are (of course!) free to look at additional references. However, we believe that the successful students will be those who spend their prep time **doing** and **practicing**, not passively reading and not memorizing canned solutions.

- The cited references also provide a basis for notation in those areas where notation varies across references.

Exam Format

- Exam time: 6 hours

  The exam will be designed with the expectation that it can be completed within four hours. However, students will be given 6 hours to complete the exam.

- Students must successfully complete (i.e., pass) four of the following six areas:
  1. Algorithms
  2. Programming/Data Structures
  3. Operating Systems
  4. Programming Languages
  5. Architecture/Computer Organization
  6. Digital Logic Circuit Design

  Note: Students may attempt more than four of the areas.

- Each area has one or more questions associated with it.

  The faculty members who write/grade the questions for a given area determine what is necessary for "successful completion" of the area.

Scope Statements for the CISE QE1

Algorithms

PG = **"Problems on Algorithms"** by Ian Parberry and William Gasarch
CLRS = "Intro to Algorithms, 3/e" by Cormen, Leiserson, Rivest, and Stein
DPV = "Algorithms" by Dasgupta, Papadimitriou, and Vazirani

Asymptotic notation
-- PG Chapter 3, CLRS Chapter 3, DPV Section 0.3.
Heaps
-- PG Section 11.1, CLRS Chapter 6, DPV Section 4.5.
Basic Graph Algs
-- PG Section 2.10, CLRS Sections 22.{1-4}
Dynamic Programming
-- PG Chapter 8, CLRS Chapter 15, DPV Chapter 6
Greedy Algorithms
-- PG Chapter 9, CLRS Chapter 16, DPV Chapter 5
Fundamental Graph Algorithms
-- PG Sections 2.10, 2.11, 7.7, 8.4, 9.2, 9.3, 9.4, and pages 28-34 in the Supplemental Proble
-- DPV Chapters 3 and 4 and Sections 4.4, 5.1, 6.1, and 6.6
-- CLRS Chapters 22, 23, 24, and 25

Programming/Data Structures

Main = *Data Structures and Other Objects Using Java, 3/e*, by M. Main, Addison-Wesley, 2006.
Sedg = *Algorithms in C++, Parts 1-4: Fundamentals, Data Structure, Sorting, Searching, 3/e*, by R. Sedgewick Addison-Wesley. 1998.
Knuth = *The Art of Computer Programming, Volume 1, 3/e*, Donald Knuth

- Performance Analysis
  (Sedg chapter 2, Knuth section 1.2.10)
- Vectors, lists, stacks, queues, and heaps
  (Sedg chapters 3-9 or Main chapters 4, 6, and 7 or Knuth section 2.2)
- Trees, Tables, Graphs, and Hashing
  (Sedg chapters 10-14 or Main chapters 9-11 or Knuth sections 2.3 and 2.4)

NOTES:
a. Answers to programming questions should be in one of *Java, C++, ML,* or *Haskell.*
b. Minor syntactic problems with answers will be ignored.
c. The Knuth reference is included primarily as a source of challenging problems.

Operating Systems

Texts
Modern Operating Systems, 3/e. Andrew Tanenbaum, Prentice Hall, 2007. ISBN: 0136006639. [MOS]

Important concepts

Processes and Thread Management, Scheduling, Concurrency, Memory Management, Interrupts, File Systems, and I/Os. (MOS: Chapters 1, 2, 3, 5, and 6. In addition, Sections 4.3, 4.4, and 4.5 are important.)

Programming Languages

FSPL == *The Formal Semantics of Programming Languages: An Introduction*, by Glynn Winskel, The MIT Press, 1993.

LNSPL == *Lecture Notes on Semantics of Programming Languages*, by Guy McCusker (based on Matthew Hennessy's course), unpublished, 2004. Available via http://www.scss.tcd.ie/Matthew.Hennessy/slexternal/resources/Guy.pdf.

Important concepts

- You should be familiar with the key notions behind operational semantics: abstract syntax, syntax-directed semantics, transition systems, evaluation, configurations, derivations, states/stores, environments, inductively defined rules, induction on derivations.

- Examples of Operational Semantics (all from FSPL):

  -- Simple imperative language (Chap 2)
  -- Simple functional language (Chap 9: Sections 9.1, 9.2, 9.5; Chap 11: Sections 11.1, 11.2, 11.5, 11.6)
  -- Nondeterminism and parallelism (Chap 14: Sections 14.1 thru 14.5)

  Important note: You are not expected to memorize all these semantics. Rather, they provide a variety of examples of the use of operational semantics. With those semantics in front of you, you should be able to complete the problems in the text. Further description of what you are expected to do is described below.

- Structural Induction (both FSPL and LNSPL):
  -- Induction on structure of terms (FSPL, Section 3.2)
  -- Induction on derivations (FSPL, Section 3.4)
  -- Definitions by induction (FSPL, Section 3.5)

  FSPL gives formal descriptions of these inductive techniques. You should also check out LNSPL for a perhaps easier-to-read description of these techniques. Furthermore, the approach used in LNSPL is sufficient for anything induction-related asked on the qual.

Expectations:

When given an operational semantics for a small language, you should be able to do the following:

-- Give the transition sequence from one specific configuration to another one.

-- Provide a derivation for a specified valid transition.

-- Add additional rules to account for new constructs to the language or to modify the behavior of existing constructs.

-- Use induction to prove that a given property holds of the language.

For more practice:

FSPL and LNSPL contain several problems throughout. If you're interested in more problems (or more examples of operational semantics), the following two

sets of notes (available via the web) are well-written and contain lots of exercises:

(1) Gordon D. Plotkin. A Structural Approach to Operational Semantics. (1981) Tech. Rep. DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark. Reprinted with corrections in J. Log. Algebr. Program. 60-61: 17-139 (2004).

[SU's library has an electronic subscription to the journal.]

(2) Matthew Hennessy. Semantics of Programming Languages. Wiley, 1990.

[The book is no longer in print, but electronic notes based on the book can be obtained via http://www.cogs.susx.ac.uk/users/matthewh/semnotes.ps.gz .]

Architecture/Computer Organization

Topics
Data and control flow, microprocessors, instruction sets, memory hierarchy, arithmetic operations, pipelining.

Sample textbook:
Computer Organization and Design, David A. Patterson and John L. Hennessy, Morgan Kaufmann, 2005, 3rd Ed. ISBN: 0006895441 (Note that this is different from the graduate-level textbook Computer Architecture - A Quantitative Approach written by the same authors.)

Focus chapters and sections:

Instructions (Chapter 2: Sections 2.2 - 2.5)
Arithmetic for computers (Chapter 3: Sections 3.2 - 3.6)
The processor: datapath and control (Chapter 5: Sections 5.2 - 5.7)
Pipelining (Chapter 6: Sections 6.2 - 6.6)
Memory hierarchy (Chapter 7: Sections 7.2 - 7.5)

Digital Logic Circuit Design

Scope:
Basic logic design concepts, flip-flops, storage devices, register, counters, design and minimization of combinational and sequential circuits.

Sample textbook
Digital Design Principles and Practices 4/e, John F. Wakerly, Prentice Hall, 2005.

Focus chapters:

Number Systems and Codes (Chapter 2)
Combinational Logic Design Principles and Practices (Chapters 4 and 6)
Sequential Logic Design Principles and Practices. (Chapters 7 and 8)

Pointers to Scopes of Previous Years

Scope statements for **AY 2010-2011, AY 2009-2010, AY 2008-2009,**

Last modified: Wed Aug 17 15:53:39 EDT 2011
Susan Older : *sueo@ecs.syr.edu*