Purpose of QE1

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 eight areas the QE 1 covers.

- Each of the eight areas has some 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 eight areas:
  1. Algorithms
  2. Programming/Data Structures
  3. Operating Systems
  4. Programming Languages
  5. Architecture/Computer Organization
  6. Digital Logic Circuit Design
  7. (New for 2020): Math Fundamentals (Discrete Math/Logic)

- 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

Algorithms:

References:

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

Topics:

1. Asymptotic notation (PG Chapter 3; CLRS Chapter 3; DPV Section 0.3)
2. Heaps (PG Section 11.1; CLRS Chapter 6; DPV Section 4.5)
3. Basic Graph Algs (PG Section 2.10; CLRS Sections 22.14)
4. Dynamic Programming (PG Chapter 8; CLRS Chapter 15; DPV Chapter 6)
5. Amortized Analysis (CLRS Chapter 17)
6. 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 Problems; DPV Chapters 3 and 4 and Sections 4.4, 5.1, 6.1, and 6.6; CLRS Chapters 22, 23, 24, and 25)
7. Greedy Algorithms (PG Chapter 9; CLRS Chapter 16; DPV Chapter 5)

Programming/Data Structures

References:

- Main = Data Structures and Other Objects Using Java, 3/e, by M. Main, Addison-Wesley, 2006.
- Knuth = The Art of Computer Programming, Volume 1, 3/e, Donald Knuth

Topics:

1. Performance Analysis (Sedg chapter 2; Knuth section 1.2.10)
2. Vectors, lists, stacks, queues, and heaps (Sedg chapters 39; Main chapters 4, 6, and 7; Knuth section 2.2)
3. Trees, Tables, Graphs, and Hashing (Sedg chapters 1014; Main chapters 911; Knuth sections 2.3 and 2.4)

Notes:

1. Answers to programming questions should be in one of Java, C++, ML, or Haskell.
2. Minor syntactic errors with answers will be ignored.
3. The Knuth reference is included primarily as a source of challenging problems.
4. The website Programming Praxis [http://programmingpraxis.com](http://programmingpraxis.com) is good source of problems at all levels of hardness.
Operating Systems

References:


Topics:

1. 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

References:


Important concepts

1. 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.

2. Examples of Operational Semantics (all from FSPL):
   (a) Simple imperative language (Chap 2)
   (b) Simple functional language (Chap 9: Sections 9.1, 9.2, 9.5; Chap 11: Sections 11.1, 11.2, 11.5, 11.6)
   (c) 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.

3. Structural Induction (both FSPL and LNSPL):
   (a) Induction on structure of terms (FSPL, Section 3.2)
   (b) Induction on derivations (FSPL, Section 3.4)
   (c) 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:

1. Give the transition sequence from one specific configuration to another one.
2. Provide a derivation for a specified valid transition.
3. Add additional rules to account for new constructs to the language or to modify the behavior of existing constructs.
4. 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:

   Computer Science Department, Aarhus University, Aarhus, Denmark. Reprinted with corrections in J. Log.
   [SU’s library has an electronic subscription to the journal.]

   [The book is no longer in print, but electronic notes based on the book can be obtained via

---

Computer Architecture

*Topics:*

1. Data and control flow
2. Microprocessors
3. Instruction sets
4. Memory hierarchy
5. Arithmetic operations
6. Pipelining

*Reference:* 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:*

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

---

Digital Logic Circuit Design

*Topics:*

1. Basic logic design concepts
2. Flip-flops
3. Storage devices
4. Register,
5. Counters
6. Design and minimization of combinational and sequential circuits
7. Register-transfer level design

*References:*

Focus chapters (in Wakerly):
1. Number Systems and Codes (Chapter 2)
2. Combinational Logic Design Principles and Practices (Chapters 4 and 6)
3. Sequential Logic Design Principles and Practices. (Chapters 7 and 8)

Probability and Statistics

References:
SR Sheldon Ross’s A First Course in Probability, Pearson

Topics:
1. Probability: be able to describe axioms of probability, possible outcomes, and events; compute probabilities and conditional probabilities, using Bayes Rule and Law of Total Probability when necessary; be able to compute and derive properties of various discrete and continuous distributions; be able to compute properties of joint distributions. (Chapters 2-6 in SR; Chapters 2-5, and 9 in FMD)
2. Basic Simulation concepts: be able to conduct simulations of continuous random variables, discrete distributions, and perform variance reduction (Chapter 6 of FMD; Chapter 10 of SR)
3. Basic Statistical concepts: be able to compute and derive properties of Mean, Variance, Covariance, Correlation, and the like as outlined in the chapters (Chapter 7,10 of FMD; Chapter 4 and 7 of SR)
4. Data summary: be able to provide (i) graphical, e.g., scatterplots, and (ii) numerical summaries, e.g., box-and-whisker plots, and explain the outcomes )Chapter 15, 16 in FMD)
5. Law of Large Numbers and Central Limit Theorem: be able to obtain Chebyshev inequality and obtain bounds on sum of random variables (Chapter 13 and 14 of FMD; Chapter 8 of SR)
6. Bootstrap and Unbiased Estimators: understand how to conduct bootstrap experiments, and obtain unbiased estimators (Chapters 18 and 19 FMD )

Math Fundamentals


1. Sets, Basic Logic, and Counting (Chapter 2, Chapter 3):
   (a) Translate between informal (but precise) English and the language of set theory.
   (b) Understand and use basic set theory notations and terminology, including set-builder notation, set membership, subset relationships, power sets, union, intersection, partition, complement, etc.
   (c) Specify and manipulate basic mathematical objects, including sets, multisets, and lists.
   (d) Understand and use logical quantifiers.
   (e) Express statements using Boolean algebra. Evaluate and prove (or disprove) the truth of such statements.
   (f) Count sets and lists, including subsets, power sets, partitions, permutations, etc. Understand and apply the binomial coefficient.
   (g) When given a set, determine whether it is countable and prove that answer.
2. Functions and Relations (Chapter 5):
   (a) Understand and use definitions relating to functions, including the term function itself, domain, range, image, inverse functions, one-to-one, onto, and bijections.
   (b) Specify and manipulate basic mathematical objects, including functions and relations.

3. Proof Techniques (Chapter 1, Chapter 4):
   (a) Express mathematical statements from the above topics in a rigorous manner.
   (b) Use mathematical and structural induction to prove mathematical properties about a variety of discrete structures.
   (c) When given a property about sets, functions, or relations, determine its validity and provide either a rigorous proof or a counter-example.
   (d) Construct and verify mathematical proofs. Select an appropriate proof technique, including direct proofs, proofs by contradiction, existential proofs, proofs by induction, and proofs by contrapositive.