CSE 300-level Course Guide

Introduction

This document serves as an informal guide to the 300-level Allen School courses. The purpose is to give new major students more details about how these courses are structured, which to take together, etc.

Disclaimer:

Not all CSE 300-level courses are listed here. Information in this document is subject to change and may not be up to date. Students should always check official course websites for up to date information.

Helpful links:

Degree Requirements
CSE Course Offerings and Time Schedule
Suggested Course Pathways

Course pathway visualization:
Tips on Choosing Courses

- There is no one set pathway in which to take all of these courses. The only course you should take ASAP is 311, since it is a prerequisite for lots of other important courses.
- Typically, 300 level courses can be divided into project-based courses and concept-based courses. Project courses, such as 332 and 331, will contain larger assignments that may span multiple weeks, taking up a large amount of time. Concept courses, such as 311 and 312, will be more about learning certain theories of computing, with problem sets to enhance your learning. These distinctions are pretty loose; every course can change, but keep them in mind when deciding which courses to pair with one another.
- Taking about two CSE courses at once is considered normal; taking more than this at once may cause you to not understand the material as well. Please think hard if you are loading your schedule with STEM courses, and as always, come talk to advisors if you have questions.

Course List

**CSE 311: Foundations of Computing I (4)**

**Summary:** Examines fundamentals of logic, set theory, induction, and algebraic structures with applications to computing; finite state machines; and limits of computability.

**Considerations:** Unlike CSE 143 and many other CSE courses, this course involves virtually no coding (usually) and heavily revolves around concepts of logic. As a result, it is typically regarded as one of the more challenging CSE courses since it requires students to critically think in new ways. Students often find that forming study groups and collaborating with others in the labs help them in this course. Recommended to take in first quarter or early on since it is a prerequisite for many 300-level and 400-level courses.

**Course Structure:** Challenging weekly problem sets that revolve around theory and logic. One midterm and one final exam.

**CSE 312: Foundations of Computing II (4)**

**Summary:** Examines fundamentals of enumeration and discrete probability; applications of randomness to computing; polynomial-time versus NP; and NP-completeness.
**Considerations**: Follows some of the concepts learned in 311, but introduces more material on statistics and probability. Usefulness of material becomes particularly apparent in upper division courses, such as AI and Machine Learning.

**Course Structure**: Weekly problem sets that revolve around discrete probability and statistics (math problems that require more conceptual thinking before solving). Occasionally a coding assignment may be substituted in as well. One midterm and one final exam.

**CSE 351: The Hardware/Software Interface (4)**

**Summary**: Examines key computational abstraction levels below modern high-level languages; number representation, assembly language, introduction to C, memory management, the operating-system process model, high-level machine architecture including the memory hierarchy, and how high-level languages are implemented.

**Considerations**: As the title suggests, this course bridges the knowledge gap between how the hardware and software of a computer interact. Concepts from this course, while seemingly foreign and sometimes confusing at first, ultimately become second nature computer science knowledge and prove extremely useful in later CSE courses and industry. Recommended to take early on.

**Course Structure**: Roughly five labs (examining assembly code and completing programs written in C) and several homework assignments (questions from course textbook) intertwined throughout the quarter. One midterm and one final exam.

**CSE 391: System and Software Tools (1)**

**Summary**: Introduction to tools commonly used in software development. Topics include using a command-line interface, writing scripts for file and string manipulation, managing user permissions, manipulating text with regular expressions, using build-management tools, and using version-control systems.

**Considerations**: Learning the terminal is extremely useful for many later CSE courses and industry. Recommended to take early on and/or concurrently with CSE 351.

**Course Structure**: Weekly homework assignments that revolve around learning and using the command line interface. No midterm or final exam, one “final assignment” (slightly longer homework). This course is graded credit/no credit (pass fail).

**CSE 331: Software Design and Implementation (4)**

**Summary**: Explores concepts and techniques for design and construction of reliable and maintainable software systems in modern high-level languages; program structure and design; program-correctness approaches, including testing; and event-driven programming (e.g., graphical user interface). Includes substantial project experience.
Considerations: Explores the Java language in more depth (good transition from CSE 143). Students typically praise the course for providing useful skills for internships, such as program design and unit testing. Although not a prerequisite for other CSE 300-level courses, this course is required for all computer science students (elective for computer engineering students). Indirectly touches on some CSE 332 material which could ease the transition into that class, however CSE 331 is not a prerequisite for that class and has a different focus. Recommended to take early on or within first year in department if possible.

Course Structure: Roughly two written homework assignments early on (related to assertions, proofs, and loop development), and then longer, individual programming projects in Java. Projects require prior planning and design, and continually build off each other (essentially designing and implementing a graph which can be eventually used to map the UW campus). Occasional reading quizzes may be given as well. One midterm and one final exam.

CSE 332: Data Structures and Parallelism (4)

Summary: Covers abstract data types and structures including dictionaries, balanced trees, hash tables, priority queues, and graphs; sorting; asymptotic analysis; fundamental graph algorithms including graph search, shortest path, and minimum spanning trees; concurrency and synchronization; and parallelism.

Considerations: Typically taken ASAP (the quarter after 311). Data structures and algorithms are fundamental concepts of computer science. From learning about sorting and searching algorithms to concepts of space and time complexity, this course is extremely useful for learning to write more efficient code and preparing for interviews and internships. Typically regarded as one of the more challenging 300-level courses along with CSE 311.

Course Structure: Roughly three programming projects involving implementing various data structures and algorithms in Java (in the last project, students implement their own chess bot and also explore concepts of concurrency). Projects are extensive and usually completed in pairs. They also consist of post-assignment write-ups where students are tasked with analyzing the efficiency and other aspects of their code. Several written exercises are also given throughout the quarter related to concepts from lecture. One midterm and one final exam.

CSE 333: Systems Programming (4)

Summary: Includes substantial programming experience in languages that expose machine characteristics and low-level data representation (e.g., C and C++); explicit
memory management; interacting with operating-system services; and cache-aware programming.

**Considerations:** Extremely useful course for learning lower level languages and some basic operating systems and network concepts. This course is a prerequisite for virtually all upper division courses that involve systems programming (operating systems, networks, distributed, etc.).

**Course Structure:** Weekly graded, take-home, programming "exercises" (usually one per lecture) usually based on content from lecture. Some can be tricky but for the most part not meant to be overly time consuming. Roughly four homework assignments which are essentially long programming projects in C (usually involving filling in portions of a large pre-existing code base). One midterm and one final exam.

**CSE 341:** Programming Languages (4)

**Summary:** After taking the course, students are better at using advanced features in various languages effectively and often say they are able to pick up new programming languages relatively quickly and effectively because they can recognize fundamental concepts and design trade-offs in any language. Can be taken at any time in the major (first quarter through last quarter), so a good course to slot in when you can.

**Considerations:** This course teaches students about the fundamentals and design behind several different programming languages. After taking the course, students gain an understanding about when to use certain programming languages over others, and often say they are able to pick up new programming languages relatively quickly and effectively.

**Course Structure:** One programming assignment per week, collectively covering several different programming languages. One midterm and one final exam.

**CSE 344:** Introduction to Data Management (4)

**Summary:** Introduces database management systems and writing applications that use such systems; data models (e.g., relational, semi-structured), query languages (e.g., SQL, XQuery), language bindings, conceptual modeling, transactions, security, database tuning, data warehousing, parallelism, and web-data management.

**Considerations:** The course begins with relatively straight-forward concepts about SQL queries and syntax. As the course progresses, it becomes increasingly more theoretical and interesting, focusing on more challenging concepts like relational algebra and design theory. This course is an excellent for those interested in database management or those interested in taking the upper division course database internals.

**Course Structure:** Roughly one homework assignment per week. Several online quizzes related to lecture content also given throughout quarter. One midterm and one final exam.
**CSE 369: Introduction to Digital Design**

**Summary:** Introduces the implementation of digital logic and its specification and simulation. Covers Boolean algebra; combinational circuits including arithmetic circuits and regular structures; sequential circuits including finite state-machines; and use of field-programmable gate arrays (FPGAs). Emphasizes simulation, high-level specification, and automatic synthesis techniques.

**Considerations:** This course begins quite slowly, assuming no prior knowledge of Verilog and digital design. As things progress, assignments become less about following steps to design a circuit and more about creatively designing and implementing the projects. Final project will take a lot of time and be much more open-ended than previous assignments.

**Course Structure:** Roughly one homework assignment or project per 1.5 weeks. Several short exams in class to understand lecture content. The second half of the course will be significantly more work than the first.

**CSE 371: Design of Digital Circuits and Systems (5)**

**Summary:** Provides a theoretical background in, and practical experience with, tools, and techniques for modeling complex digital systems with the Verilog hardware description language, maintaining signal integrity, managing power consumption, and ensuring robust intra- and inter-system communication.

**Considerations:** This is one of the more hardware-based courses in our CSE curriculum; in fact, this course is offered jointly with EE 371. You should have a firm grasp of the digital design-type information such as that offered in CSE 369.

**Course Structure:** There will be twice-weekly lectures on the course material, then weekly labs in which you work on that week’s homework assignment or project. Course culminates with a final project.