School of Computer Science Courses

Note on Course Numbers

Each Carnegie Mellon course number begins with a two-digit prefix which designates the department offering the course (76-xxx courses are offered by the Department of English, etc.). Although each department maintains its own course numbering practices, typically the first digit after the prefix indicates the class level: xx-1xx courses are freshmen-level, xx-2xx courses are sophomore level, etc. xx-6xx courses may be either undergraduate senior-level or graduate-level, depending on the department. xx-7xx courses and higher are graduate-level. Please consult the Schedule of Classes (https://enr-apps.as.cmu.edu/open/SOC/SOCServlet) each semester for course offerings and for any necessary pre-requisites or co-requisites.

15-050 Study Abroad
All Semesters
Students who are interested in studying abroad should first contact the Office of International Education. More information on Study Abroad is available on OIE’s Study Abroad page and at the CS Undergraduate Office.

15-075 Computer Science Co-Op
All Semesters
Students who are interested in a Co-Op experience with an external employer typically do so in their junior year. A Co-Op is distinguished from a summer internship in that it encompasses a summer and a contiguous semester, either Spring-Summer or Summer-Fall. A list of companies who are interested in hiring Co-Op students is available from the SCS Career Consultant at the Career Center. More information on the Computer Science Co-Op program is available at the CS Undergraduate Office.

15-090 Computer Science Practicum
All Semesters: 3 units
This course is for international students who are interested in working for Curricular Practical Training (CPT). Such students interested in CPT must first be authorized by the Office of International Education before being able to enroll in the Practicum course. More information on CPT is available on OIE’s Foreign Student Employment page and at the CS Undergraduate Office.

15-110 Principles of Computing
All Semesters: 10 units
A course in fundamental computing principles for students with minimal or no computing background. Programming constructs: sequencing, selection, iteration, and recursion. Data organization: arrays and lists. Use of abstraction in computing: data representation, computer organization, computer networks, functional decomposition, and application programming interfaces. Use of computational principles in problem-solving: divide and conquer, randomness, and concurrency. Classification of computational problems based on complexity, non-computable functions, and usable heuristics to find reasonable solutions to complex problems. Social, ethical and legal issues associated with the development of new computational artifacts will also be discussed.

15-112 Fundamentals of Programming and Computer Science
All Semesters: 12 units
A technical introduction to the fundamentals of programming with an emphasis on producing clear, robust, and reasonably efficient code using top-down design, informal analysis, and effective testing and debugging. Starting from first principles, we will cover a large subset of the Python programming language, including its standard libraries and programming paradigms. We will also target numerous deployment scenarios, including standalone programs, shell scripts, and web-based applications. This course assumes no prior programming experience. Even so, it is a fast-paced and rigorous preparation for 15-122. Students seeking a more gentle introduction to computer science should consider first taking 15-110. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course.

15-121 Introduction to Data Structures
Fall and Spring: 10 units
A continuation of the process of program design and analysis for students with some prior programming experience (functions, loops, and arrays, not necessarily in Java). The course reinforces object-oriented programming techniques in Java and covers data aggregates, data structures (e.g., linked lists, stacks, queues, trees, and graphs), and an introduction to the analysis of algorithms that operate on those data structures. Prerequisites: 15-112 or 15-110.

15-122 Principles of Imperative Computation
All Semesters: 10 units
For students with a basic understanding of programming (variables, expressions, loops, arrays, functions). Teaches imperative programming and methods for ensuring the correctness of programs. Students will learn the process and concepts needed to go from high-level descriptions of algorithms to correct imperative implementations, with specific application to basic data structures and algorithms. Much of the course will be conducted in a subset of C amenable to verification, with a transition to full C near the end. This course prepares students for 15-213 and 15-210. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course. Prerequisite: 15-112 Corequisite: 21-127.

Course Website: http://www.andrew.cmu.edu/user/annpenney/15122-m13/webby/home.html

15-128 Freshman Immigration Course
Fall: 1 unit
This Freshman Immigration Course is taken by freshman Computer Science majors on the Pittsburgh campus. The course is designed to acquaint incoming majors with computer science at CMU. Talks range from historical perspectives in the field to descriptions of the cutting edge research being conducted in the School of Computer Science. 15-128 is a one unit course and is graded pass/fail. Enrollment is limited to SCS Freshmen ONLY.

Course Website: http://www.andrew.cmu.edu/course/15-128/

15-129 Freshman Immigration Course II
Fall: 3 units
This course is ONLY offered at Carnegie Mellon in Qatar. Students and instructors will solve different problems each week by searching the Web and other likely places for answers. The problems will be submitted by other faculty who will grade the quality of the answers. Students will learn strategies and techniques for finding information on the Web more efficiently; learn when to start with a search engine, a subject-oriented directory, or other tools; explore and practice using advanced search syntax for major search engines; experience specialized search engines for images, sound, multimedia, newsgroups, and discussion lists as well as subject-specific search engines; discover valuable resources to help keep you up-to-date in this fast-changing environment.

15-131 Great Practical Ideas for Computer Scientists
Fall: 2 units
THIS COURSE IS OPEN TO CS FRESHMAN ONLY. Throughout your education as a Computer Scientist at Carnegie Mellon, you will take courses on programming, theoretical ideas, logic, systems, etc. As you progress, you will be expected to pick up the so-called "tools of the trade." This course is intended to help you learn what you need to know in a friendly, low-stress, high-support way. We will discuss UNIX, LaTeX, debugging and many other essential tools. Laptop required. (Laptops will be available for those without their own laptops.)

15-150 Principles of Functional Programming
Fall and Spring: 10 units
An introduction to programming based on a "functional" model of computation. The functional model is a natural generalization of algebra in which programs are formulas that describe the output of a computation in terms of its inputs—that is, as a function. But instead of being confined to real- or complex-valued functions, the functional model extends the algebraic view to a very rich class of data types, including not only aggregates built up from other types, but also functions themselves as values. This course is an introduction to programming that is focused on the central concepts of function and type. One major theme is the role of types in structuring large programs into separate modules, and the integration of imperative programming through the introduction of data types whose values may be altered during computation. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course. Prerequisites: (21-127 or 15-131) and 15-112.
15-210 Parallel and Sequential Data Structures and Algorithms
Fall and Spring: 12 units
Teaches students about how to design, analyze, and program algorithms and data structures. The course emphasizes parallel algorithms and analysis, and how sequential algorithms can be considered a special case. The course goes into more theoretical content on algorithm analysis than 15-122 and 15-150 while still including a significant programming component and covering a variety of practical applications such as problems in data analysis, graphics, text processing, and the computational sciences. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course. Prerequisites: 15-150 and 15-122.

15-211 Fundamental Data Structures and Algorithms
Fall and Spring: 12 units
Fundamental programming concepts are presented together with supporting theoretical foundations and practical applications. This course emphasizes the practical application of techniques for writing and analyzing programs: data abstraction, program verification, and performance analysis. These techniques are applied in the design and analysis of fundamental algorithms and data structures. The course is currently taught in Java. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course. Prerequisites: 21-127 and 15-121.

15-213 Introduction to Computer Systems
Fall and Spring: 12 units
This course provides a programmer’s view of how computer systems execute programs, store information, and communicate. It enables students to become more effective programmers, especially in dealing with issues of performance, portability, and robustness. It also serves as a foundation for courses on compilers, operating systems, and computer architecture, where a deeper understanding of systems-level issues is required. Topics covered include: machine-level code and its generation by optimizing compilers, performance evaluation, and optimization, computer arithmetic, memory organization and management, and supporting concurrent computation. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course. Prerequisite: 15-122.

15-214 Principles of Software Construction: Objects, Design, and Concurrency
Fall and Spring: 12 units
Software engineers today are less likely to design data structures and algorithms from scratch and more likely to build systems from library and framework components. In this course, students engage with concepts related to the construction of software systems at scale, building on their understanding of the basic building blocks of data structures, algorithms, program structures, and computer structures. The course covers technical topics in four areas: (1) concepts of design for complex systems, (2) object-oriented programing, (3) static and dynamic analysis for programs, and (4) concurrent and distributed software. Student assignments involve engagement with complex software such as distributed massively multiplayer game systems and frameworks for graphical user interaction. Prerequisites: (15-121 or 15-122) and (21-127 or 15-151).

15-221 Technical Communication for Computer Scientists
All Semesters: 9 units
The course is designed for sophomore computer science majors to improve their abilities in practical, professional communications (both written and oral). It aims to help students compose clear, concise technical writings and oral presentations for multi-level audiences. Assignments include technical definitions, descriptions, instructions, process explanations, abstracts, memos, and research reports. Assignments may incorporate recent computer science research at Carnegie Mellon, projects in related technical courses, and professional case studies. Sophomores will likely find the course more useful if they have either had an internship or faculty-supervised research, including SURG projects prior to enrollment. Prerequisite: 76-101.

15-237 Special Topic: Cross-Platform Mobile Web Apps
12 units
An introduction to writing cross-platform mobile web apps. Using a tool chain based on HTML5, CSS3, JavaScript, and a variety of supporting frameworks, we will write apps that are effectively designed both for desktop and mobile browsers, and which can be converted into native apps for Android, iOS, and Windows Phone 7 devices. Additional topics will include designing user interfaces for mobile devices, accessing mobile device API’s (such as accelerometer, GPS, compass, or camera), and power management issues. While this course focuses on browser-side technologies, we will briefly explore JavaScript-based server-side technologies (though students should consider 15-437 for extensive treatment of server-side topics). Note that we will not be writing native apps for Objective-C for iOS nor in Java for Android, though we may include some brief exposure to these technologies near the end of the course. Prerequisite: 15-112.

15-251 Great Theoretical Ideas in Computer Science
Fall and Spring: 12 units
This course is about how to use theoretical ideas to formulate and solve problems in computer science. It integrates mathematical material with general problem solving techniques and computer science applications. Examples are drawn from Algorithms, Complexity Theory, Game Theory, Probability Theory, Graph Theory, Automata Theory, Algebra, Cryptography, and Combinatorics. Assignments involve both mathematical proofs and programming. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course. Prerequisites: 15-112 and (21-127 or 15-151). Course Website: http://www.cs.cmu.edu/~15251

15-295 Competition Programming and Problem Solving
Fall and Spring: 6 units
Each year, Carnegie Mellon fields two teams for participation in the ACM-ICPC Regional Programming Contest. During many recent years, one of those teams has earned the right to represent Carnegie Mellon at the ACM-ICPC World Finals. This course is a vehicle for those who consistently and rigorously train in preparation for the contests to earn course credit for their effort and achievement. Preparation involves the study of algorithms, the practice of programming and debugging, the development of test sets, and the growth of team, communication, and problem solving skills. Neither the course grade nor the number of units earned are dependent on ranking in any contest. Students are not required to earn course credit to participate in practices or to compete in ACM-ICPC events. Consistent, disciplined participation in team practices and associated individual preparation earns three (3) units of credit. Six (6) units of credit are possible for those who routinely complete significant homework and/or teamwork assignments outside of normal group meetings. Students interested in the course should attend during the first week of classes to discuss enrollment details. Prerequisite: 15-122.

15-312 Foundations of Programming Languages
Spring: 12 units
This course discusses in depth many of the concepts underlying the design, definition, implementation, and use of modern programming languages. Formal approaches to defining the syntax and semantics are used to describe the fundamental concepts underlying programming languages. A variety of programming paradigms are covered such as imperative, functional, logic, and concurrent programming. In addition to the formal studies, experience with programming in the languages is used to illustrate how different design goals can lead to radically different languages and models of computation. Prerequisites: (15-210 and 15-251) or 15-212.

15-313 Foundations of Software Engineering
Fall: 12 units
Students gain exposure to the fundamentals of modern software engineering. This includes both core CS technical knowledge and the means by which this knowledge can be applied in the practical engineering of complex software. Topics related to software artifacts include design models, patterns, coding, static and dynamic analysis, testing and inspection, measurement, and software architecture and frameworks. Topics related to software process include modeling, requirements engineering, process models and evaluation, team development, and supply chain issues including outsourcing and open source. This course has a strong technical focus, and will include both written and programming assignments. Students will get experience with modern software engineering tools. Prerequisite: 15-214.
15-317 Constructive Logic
Fall: 9 units
This multidisciplinary junior-level course is designed to provide a thorough introduction to modern constructive logic, its roots in philosophy, its numerous applications in computer science, and its mathematical properties. Some of the topics to be covered are intuitionistic logic, inductive definitions, functional programming, type theory, realizability, connections between classical and constructive logic, decidable classes. This course counts as a Fundamentals course in the Computer Science major.
Prerequisites: 15-210 or 15-212.

15-319 Cloud Computing
9 units
This on-line course gives students an overview of the field of Cloud Computing, its enabling technologies, main building blocks, and hands-on experience through 4 projects utilizing a public cloud (Amazon Web Services). Cloud computing services are being adopted widely across a variety of organizations and in many domains. Simply, cloud computing is the delivery of computing as a service over a network, whereby distributed resources are rented, rather than owned, by an end user as a utility. The course will introduce this domain and cover the topics of data centers, resource management, cloud storage, and programming models. As an introduction, we will discuss the motivating factors, benefits, challenges, and service models. Modern data centers enable many of the economic and technological benefits of the cloud paradigm; hence, we will describe several concepts behind data center design and management. Next, we will focus on existing approaches for resource management including resource abstraction, sharing and virtualization technologies. Students will learn about different cloud storage concepts including data distribution, durability, consistency and redundancy. HDFS, PVFS, HBase, Cassandra and S3 will be presented as examples of underlying distributed file systems. Students will understand the details of the MapReduce programming model and gain a broad overview of alternative programming models such as Pregel, and GraphLab. Students will work with a public cloud service (such as Amazon Web Services), use it to rent and provision compute resources and then program and deploy applications that run on these resources. Students will work with cloud storage systems and learn to develop applications in the MapReduce programming paradigm.
Prerequisite: 15-213.

15-321 Research Methods for Experimental Computer Science
Fall: 9 units
The success or failure of an experiment can turn on the details of how the experiment was performed — the experimental method. It is critical that the methodology be consistent with whatever research hypothesis is being pursued. The goal of this project-based course is to give students familiarity with and appreciation for the subtleties of a range of experimental techniques essential to high-quality empirical research. The course is based on a project that will illustrate important concepts of research methods. Example topics include user-study design and operation, data gathering, data diagnosis, experiment design and execution, signal detection, performance evaluation, error analysis, reporting results, etc. The project will explore the challenging open problem of keystroke dynamics, a biometric metric which seeks to identify/authenticate/discriminate users on the basis of their typing styles. In this context, numerous methodological issues provide the stimulus for learning how to perform empirical research from the beginning to the end of a project. Lectures will present necessary background material about the problem area and experimental methods; homework assignments and a team project will give students guided, hands-on, research and practical experience. At the end of the course, students will be acquainted with the basic issues in experimental methods, and will be able to perform independent work using the lessons learned and resources provided. Enrollment is limited to graduate students, juniors, and seniors. It will be helpful for students to have some experience with scripting languages (e.g., Perl, Python, Tcl, etc); statistically-oriented packages (e.g., R, Matlab, Weka) or other implementations of various machine-learning-type classifiers. This course can be used to satisfy the Lab requirement for the Computer Science major.
Prerequisite: 15-213.

15-322 Introduction to Computer Music
Spring: 9 units
Computers are used to synthesize sound, process signals, and compose music. Personal computers have replaced studios full of sound recording and processing equipment, completing a revolution that began with recording and electronics. In this course, students will learn the fundamentals of digital audio, basic sound synthesis algorithms, and techniques for digital audio effects and processing. Students will apply their knowledge in programming assignments using a very high-level programming language for sound synthesis and composition. In a final project, students will demonstrate their mastery of tools and techniques through music composition or by the implementation of a significant sound-processing technique.
Prerequisite: 15-112.

15-323 Computer Music Systems and Information Processing
Fall: 9 units
This course presents concepts and techniques for representing and manipulating discrete music information, both in real time and off line. Representations of music as explicitly timed event sequences will be introduced, and students will learn how to build efficient run-time systems for event scheduling, tempo control, and interactive processing. The MIDI protocol is used to capture real-time performance information and to generate sound. The course will also cover non-real-time processing of music data, including Markov models, style recognition, computer accompaniment, query-by-humming, and algorithmic composition. This course is independent of, and complementary to 15-322, Introduction to Computer Music, which focuses on sound synthesis and signal processing.
Prerequisite: 15-122.

15-348 Embedded Systems
Fall: 9 units
This course is ONLY offered at Carnegie Mellon in Qatar. This course covers the broad range of foundational skills that apply across all embedded computer system application areas, from thermostats to self-driving vehicles. The emphasis is at the layer where hardware meets software. Topics include microcontroller hardware, assembly language, embedded C programming, analog I/O, timers, code optimization, interrupts, and concurrency. Real world engineering practices, constraints, and example applications are integrated throughout the course. Weekly hands-on hardware and software experiences with an industry-strength automotive embedded controller are coordinated with the lecture content to reinforce core skills.
Prerequisite: 15-123.

15-349 Introduction to Computer and Network Security
Fall: 9 units
This course is ONLY offered at Carnegie Mellon in Qatar. This course is meant to offer Computer Science undergraduate students in their junior or senior year a broad overview of the field of computer security. Students will learn the basic concepts in computer security including software vulnerability analysis and defense, networking and wireless security, applied cryptography, as well as ethical, legal, social and economic facets of security. Students will also learn the fundamental methodology for how to design and analyze security critical systems.
Prerequisite: 15-123.

15-354 Computational Discrete Mathematics
Fall: 12 units
This course is about the computational aspects of some of the standard concepts of discrete mathematics (relations, functions, logic, graphs, algebra, automata), with emphasis on efficient algorithms. We begin with a brief introduction to computability and computational complexity. Other topics include: iteration, orbits and fixed points, order and equivalence relations, propositional logic and satisfiability testing, finite fields and shift register sequences, finite state machines, and cellular automata. Computational support for some of the material is available in the form of a Mathematica package.
Prerequisite: 15-251.

Course Website: http://www.cs.cmu.edu/~cdm/
15-355 Modern Computer Algebra
Fall: 9 units
The goal of this course is to investigate the relationship between algebra and computation. The course is designed to expose students to algorithms used for symbolic computation, as well as to the concepts from modern algebra which are applied to the development of these algorithms. This course provides a hands-on introduction to many of the most important ideas used in symbolic mathematical computation, which involves solving system of polynomial equations (via Groebner bases), analytic integration, and solving linear difference equations. Throughout the course the computer algebra system Mathematica will be used for computation.
Prerequisite: 15-251.
Course Website: http://www.andrew.cmu.edu/course/15-355/

15-359 Probability and Computing
Spring: 12 units
Probability theory has become indispensable in computer science. In areas such as artificial intelligence and computer science theory, probabilistic methods and ideas based on randomization are central. In other areas such as networks and systems, probability is becoming an increasingly useful framework for handling uncertainty and modeling the patterns of data that occur in complex systems. This course gives an introduction to probability as it is used in computer science theory and practice, drawing on applications and current research developments as motivation and context. Topics include combinatorial probability and random graphs, heavy tail distributions, concentration inequalities, various randomized algorithms, sampling random variables and computer simulation, and Markov chains and their many applications, from Web search engines to models of network protocols. The course will assume familiarity with 3-D calculus and linear algebra.
Prerequisites: 15-251 and 21-259 and 21-241.
Course Website: http://www.cs.cmu.edu/~harchol/15359/class.html

15-381 Artificial Intelligence: Representation and Problem Solving
Fall: 9 units
This course is about the theory and practice of Artificial Intelligence. We will study modern techniques for computers to represent task-relevant information and make intelligent (i.e. satisfying or optimal) decisions towards the achievement of goals. The search and problem solving methods are applicable throughout a large range of industrial, civil, medical, financial, robotic, and information systems. We will investigate questions about AI systems such as: how to represent knowledge, how to effectively generate appropriate sequences of actions and how to search among alternatives to find optimal or near-optimal solutions. We will also explore how to deal with uncertainty in the world, how to learn from experience, and how to learn decision rules from data. We expect that by the end of the course students will have a thorough understanding of the algorithmic foundations of AI, how probability and AI are closely interrelated, and how automated agents learn. We also expect students to acquire a strong appreciation of the big-picture aspects of developing fully autonomous intelligent agents. Other lectures will introduce additional aspects of AI, including natural language processing, web-based search engines, industrial applications, autonomous robotics, and economic/game-theoretic decision making.
Prerequisites: 15-122 or 15-211.

15-383 Introduction to Text Processing
Fall: 6 units
Text processing is a mini-course about text basic techniques of processing human language in text format. The course has theoretical and hands-on components. In the theoretical component, the course will discuss challenges in processing human languages, and review the basics of statistics and probability theory and their application to language problems. In the hands-on part, students will learn about Python programming and use it to process large volumes of text using various techniques. The processing will range from simple steps such as tokenization and part-of-speech tagging to full-fledged applications such as statistical machine translation, search and document/topic classification. The course is suited for junior and senior students in CS and IS.
Prerequisites: 15-121 or 15-122.

15-385 Introduction to Computer Vision
Spring: 6 units
An introduction to the science and engineering of computer vision, i.e. the analysis of the patterns in visual images with the view to understanding the objects and processes in the world that generate them. Major topics include image formation and sensing, fourier analysis, edge and contour detection, inference of depth, shape and motion, classification, recognition, tracking, and active vision. The emphasis is on the learning of fundamental mathematical concepts and techniques and applying them to solve real vision problems. The discussion will also include comparison with human and animal vision from psychological and biological perspectives. Students will learn to think mathematically and develop skills in translating ideas and mathematical thoughts into programs to solve real vision problems.
Prerequisites: (18-202 and 15-122) or (21-259 and 15-122 and 21-241).

15-386 Neural Computation
Spring: 9 units
Computational neuroscience is an interdisciplinary science that seeks to understand how the brain computes to achieve natural intelligence. It seeks to understand the computational principles and mechanisms of intelligent behaviors and mental abilities — such as perception, language, motor control, and learning — by building artificial systems and computational models with the same capabilities. This course explores how neurons encode and process information, adapt and learn, communicate, cooperate, compete and compute at the individual level as well as at the levels of networks and systems. It will introduce basic concepts in computational modeling, information theory, signal processing, system analysis, statistical and probabilistic inference. Concrete examples will be drawn from the visual system and the motor systems, and studied from computational, psychological and biological perspectives. Students will learn to perform computational experiments using Matlab and quantitative studies of neurons and neuronal networks.
Prerequisites: 21-122 and 15-112.

15-387 Computational Perception
Fall: 9 units
In this course, we will first cover the biological and psychological foundational knowledge of biological perceptual systems, and then apply computational thinking to investigate the principles and mechanisms underlying natural perception. The course will focus on vision this year, but will also touch upon other sensory modalities. You will learn how to reason scientifically and computationally about problems and issues in perception, how to extract the essential computational properties of those abstract ideas, and finally how to convert these into explicit mathematical models and computational algorithms. Topics include perceptual representation and inference, perceptual organization, perceptual constancy, object recognition, learning and scene analysis.
Prerequisites: First year college calculus, some basic knowledge of linear algebra and probability and some programming experience are desirable.
Prerequisites: 21-122 and 21-241 and 15-112.

15-390 Entrepreneurship for Computer Science
Fall: 9 units
This course is an introduction to Entrepreneurship designed to develop skills related to entrepreneurship and innovation for non-business undergraduate and graduate students in the School of Computer Science. The course assumes no background courses in business and is appropriate for those who are interested in bringing innovations to market either through new companies or existing companies. The course provides an overview of entrepreneurship and innovation, develops an entrepreneurial frame of mind, and provides a framework for learning the rudiments of how to generate ideas. Students come up with or are presented with potential ideas and learn how to develop these ideas into opportunities, and to explore their potential for becoming viable businesses. They learn how to do market research, to develop go-to-market strategies, value propositions and to differentiate their products or services from potential competitors. The focus is on understanding and developing strategies for approaching the key elements of the entrepreneurial process — opportunity, resources and team. The course consists of a balance of lectures, case studies and encounters with entrepreneurs, investors and business professionals. The students are exposed to financial and intellectual property issues, and encounter a real world perspective on entrepreneurship, innovation and leadership. The output of the course is a mini-business plan or venture opportunity screening document that can be developed into a business plan in a subsequent course entitled New Venture Creation or through independent study.
15-392 Special Topic: Secure Programming
Spring: 9 units
This course provides a detailed explanation of common programming errors in C and C++ and describes how these errors can lead to software systems that are vulnerable to exploitation. The course concentrates on security issues intrinsic to the C and C++ programming languages and associated libraries. It does not emphasize security issues involving interactions with external systems such as databases and web servers, as these are rich topics on their own. Topics to be covered include the secure and insecure use of integers, arrays, strings, dynamic memory, formatted input/output functions, and file I/O.
Prerequisite: 15-213.
Course Website: https://www.securecoding.cert.org/confluence/display/sci/15392+Secure+Programming

15-410 Operating System Design and Implementation
Fall and Spring: 12 units
Operating System Design and Implementation is a rigorous hands-on introduction to the principles and practice of operating systems. The core experience is writing a small Unix-inspired OS kernel, in C with some x86 assembly language, which runs on a PC hardware simulator (and on actual PC hardware if you wish). Work is done in two-person teams, and "team programming" skills (source control, modularity, documentation) are emphasized. The size and scope of the programming assignments typically result in students significantly developing their design, implementation, and debugging abilities. Core concepts include the process model, virtual memory, threads, synchronization, and deadlock; the course also surveys higher-level OS topics including file systems, interprocess communication, networking, and security. Students, especially graduate students, who have not satisfied the prerequisite at Carnegie Mellon are strongly cautioned - to enter the class you must be able to write a storage allocator in C, use a debugger, understand 2’s-complement arithmetic, and translate between C and x86 assembly language. The instructor may require you to complete a code assessment exercise before the first week of the semester in order to remain registered in the class. Auditing this course is usually full, and we generally receive many more requests to audit than we can accept. If you wish to audit, please have your advisor contact us before the semester begins to discuss your educational goals.
Prerequisite: 15-213.

15-411 Compiler Design
Fall: 12 units
This course covers the design and implementation of compiler and run-time systems for high-level languages, and examines the interaction between language design, compiler design, and run-time organization. Topics covered include syntactic and lexical analysis, handling of user-defined types and type-checking, context analysis, code generation and optimization, and memory management and run-time organization.
Prerequisites: 15-213 or 15-312.
Course Website: http://symbolaris.com/course/compiler12.html

15-412 Operating System Practicum
Fall
The goal of this class is for students to acquire hands-on experience with operating-system code as it is developed and deployed in the real world. Groups of two to four students will select, build, install, and become familiar with an open-source operating system project; propose a significant extension or upgrade to that project; and develop a production-quality implementation meeting the coding standards of that project. Unless infeasible, the results will be submitted to the project for inclusion in the code base. Variations on this theme are possible at the discretion of the instructor. For example, it may be possible to work within the context of a non-operating-system software infrastructure project (window system, web server, or embedded network device kernel) or to extend a 15-410 student kernel. In some situations students may work alone. Group membership and unit count (9 units versus 12) will be decided by the third week of the semester. Contributing to a real-world project will involve engaging in some mixture of messy, potentially open-ended activities such as: learning a revision control system, writing a short design document, creating and updating a simple project plan, participating in an informal code review, synthesizing scattered information about hardware and software, classifying and/or reading large amounts of code written by various people over a long period of time, etc.
Prerequisite: 15-410.

15-413 Software Engineering Practicum
Spring: 12 units
This course is modeled after the successful MSE studio and MSIT practicum experiences used in the ISR software engineering masters programs. Students will carry out a semester-long project for a real client in small teams. A few lectures at the beginning of the course will introduce the process, tools, and documents to be used in the course, all of which will be specified by the instructors. Throughout the project, groups will be mentored through weekly team meetings with faculty. Teams will make regular presentations on the following topics: risk management, project planning, requirements, architecture, detailed design, quality assurance, final product presentations, and reflections on the experience. Teams will all deliver documents on each of the above topics. Evaluation will be based on the in-class presentations, process and product documentation deliverables, how well the teams follow SE practices as judged by their mentor, and finally the client’s satisfaction with the product. Individual grades within a team will be influenced by peer reviews, individual reflection documents, mentor impressions, and presentation performance. Students will leave the course with a firsthand understanding of the software engineering realities that drive SE practices. They will have concrete experience with these practices, and will have engaged in active reflection on this experience. They will have teamwork, process, and product skills to support immediate competency in a software engineering organization, along with a deeper understanding that prepares them to evaluate the new processes and techniques they will encounter in the workplace.
Prerequisite: 15-313.

15-414 Bug Catching: Automated Program Verification and Testing
Fall: 9 units
Many CS and ECE students will be developing software and hardware that must be ultra reliable at some point in their careers. Logical errors in such designs can be costly, even life threatening. There have already been a number of well publicized errors like the Intel Pentium floating point error and the Ariane 5 crash. In this course we will study tools for finding and preventing logical errors. Three types of tools will be studied: automated theorem proving, static program analysis techniques like model checking and tools based on static program analysis. Although students will learn the theoretical basis for such tools, the emphasis will be on actually using them on real examples. This course can be used to satisfy the Logic & Languages requirement for the Computer Science major.
Prerequisites: (15-122 or 15-211) and 15-251.
Course Website: http://15440

15-415 Database Applications
Spring: 12 units
This course covers the fundamental topics for Database Management Systems: Database System Architectural Principles (ACID properties; data abstraction; external, conceptual, and internal schemata; data independence; data definition and data manipulation languages), Data model (entity-relationship and relational data models; data structures, integrity constraints, and operations for each data model; relational query languages: SQL, algebra, calculus), Theory of database design (functional dependencies; normal forms; dependency preservation; information loss), Query processing (equivalence of expressions, algebraic manipulation; optimization of selections and joins), Storage Strategies (indices, B-trees, hashing), Query Processing (execution of sort, join, and aggregation operators), and Transaction Processing (recovery and concurrency control).
Prerequisites: 15-210 or (15-211 and 15-213).
Course Website: http://www.cs.cmu.edu/~christos/courses/dbms.S13/

15-417 HOT Compilation
Intermittent: 12 units
The course covers the implementation of compilers for higher-order, typed languages such as ML and Haskell, and gives an introduction to type-preserving compilation. Topics covered include type inference, elaboration, CPS conversion, closure conversion, garbage collection, phase splitting, and typed assembly language.
Prerequisites: 15-312 or 15-317.

15-418 Parallel Computer Architecture and Programming
Spring: 12 units
The fundamental principles and engineering tradeoffs involved in designing modern parallel computers, as well as the programming techniques to effectively utilize these machines. Topics include naming shared data, synchronizing threads, and the latency and bandwidth associated with communication. Case studies on shared-memory, message-passing, data-parallel and dataflow architectures will be used to illustrate these techniques and tradeoffs. Programming assignments will be performed on one or more commercial multiprocessors, and there will be a significant course project.
Prerequisite: 15-213.
15-421 Information Security and Privacy
Fall: 12 units
The objective of this course is to introduce students to the technologies of Web Commerce, Security and Privacy as well as to related business, policy, and usability issues. Content: Over the past 15 years, the Web has become an integral part of our daily life, whether at home or at work. This course provides students with an overview of the technologies and practices associated with Web Security, Privacy and Commerce. In the process, students will learn what it takes to design and develop successful web applications and services, reconciling security, privacy, usability and business considerations. The course is organized around two parts: Part I - Web Security & Privacy Technologies The big picture, gentle introduction to cryptography, digital signatures, key management, authentication, Internet security protocols, certificates & PKI, decentralized trust management, privacy enhancing technologies, electronic payments. Part II - Web Commerce The big picture, Internet marketing & personalization, search engines, B2B and electronic markets, P2P, Web 2.0, Mobile Commerce, social networking. Format: Lectures (including guest lectures), discussions, student presentations, and class projects. Prerequisites: 15122 or 15211 Prerequisites: 15-122 or 15-211.
Course Website: http://www.cs.cmu.edu/~sadeh/15%20421.html

15-424 Special Topic: Foundations of Cyber-Physical Systems
12 units
Cyber-physical systems (CPSs) combine cyber effects (computation and/or communication) with physical effects (motion or other physical processes). Designing algorithms to control CPSs, such as those in cars, aircraft and robots, is challenging due to their tight coupling with physical behavior. At the same time, it is vital that these algorithms be correct, since we rely on CPSs for safety-critical tasks like keeping aircraft from colliding. Students in this course will understand the core principles behind CPSs, develop models and controls, identify safety specifications and critical properties of CPSs, understand abstraction and system architectures, learn how to design by invariant, reason rigorously about CPS models, verify CPS models of appropriate scale, understand the semantics of a CPS model and develop an intuition for operational effects. Students will write hybrid programs (HPs), which capture relevant dynamical aspects of CPSs in a simple programming language with a simple semantics, allowing the programmer to refer to real-valued variables representing real quantities and specify their dynamics as part of the HP. Prerequisites: 15-122 and 15-251 and 21-122.
Course Website: http://symbolaris.com/course/fcps13.html

15-437 Web Application Development
Fall and Spring: 12 units
This course will introduce concepts in programming web application servers. We will study the fundamental architectural elements of programming web sites that produce content dynamically. The primary technology introduced will be Java Servlets and Java Server Pages (JSPs), but we will cover related topics as necessary so that students can build significant applications. Such topics include: HTTP, HTML and XML, JavaBeans, Design Patterns, Tag Libraries, Relational Databases, Object-Relational Mapping tools, Security, Web Services, Frameworks, Internationalization, and Scalability and Performance Issues. This course is recommended for students contemplating enrollment in 15-413 (the Software Engineering Project), since many of the projects in 15-413 are expected to be web-based. Students must be comfortable programming in Java to register for this course. Students must provide their own computer hardware for this course. Please see the Related URL above for more information. Prerequisite: 15-214.
Course Website: http://www.jeffepinger.com/jlle/web-app.html

15-440 Distributed Systems
Spring: 12 units
The goals of this course are twofold: First, for students to gain an understanding of the principles and techniques behind the design of distributed systems, such as locking, concurrency, scheduling, and communication across the network. Second, for students to gain practical experience designing, implementing, and debugging real distributed systems. The major themes this course will teach include security, scheduling, concurrency and concurrent programming, naming, abstraction and modularity, imperfect communication and other types of failure, protection from accidental and malicious harm, optimism, and the use of instrumentation and monitoring and debugging tools and techniques. As the creation and management of software systems is a fundamental goal of any undergraduate systems course, students will design, implement, and debug large programming projects. As a consequence, competency in both the C and Java programming languages is required. Prerequisite: 15-213.

15-441 Computer Networks
Fall and Spring: 12 units
The emphasis in this course will be on the basic performance and engineering trade-offs in the design and implementation of computer networks. To make the issues more concrete, the class includes several multi-week projects requiring significant design and implementation. The goal is for students to learn not only what computer networks are and how they work today, but also why they are designed the way they are and how they are likely to evolve in the future. We will draw examples primarily from the Internet. Topics to be covered include: network architecture, routing, congestion/flow/error control, naming and addressing, peer-to-peer and the web, internetworking, and network security. Prerequisite: 15-213.

15-451 Algorithm Design and Analysis
Fall and Spring: 12 units
This course is about the design and analysis of algorithms. We study specific algorithms for a variety of problems, as well as general design and analysis techniques. Specific topics include searching, sorting, algorithms for graph problems, efficient data structures, lower bounds and NP-completeness. A variety of other topics may be covered at the discretion of the instructor. These include parallel algorithms, randomized algorithms, geometric algorithms, low level techniques for efficient programming, cryptography, and cryptographic protocols. Prerequisites: 15-251 and 15-210 and 21-241.

15-453 Formal Languages, Automata, and Computability
Spring: 9 units
An introduction to the fundamental ideas and models underlying computing: finite automata, regular sets, pushdown automata, context-free grammars, Turing machines, undecidability, and complexity theory. Prerequisites: 21-228 or 15-251.

15-455 Undergraduate Complexity Theory
Intermittent: 9 units
Complexity theory is the study of how much of a resource (such as time, space, parallelism, or randomness) is required to perform some of the computations that interest us the most. In a standard algorithms course, one concentrates on giving resource efficient methods to solve interesting problems. In this course, we concentrate on techniques that prove or suggest that there are no efficient methods to solve many important problems. We will develop the theory of various complexity classes, such as P, NP, co-NP, PH, #P, PSPACE, NC, AC, L, NL, UP, RP, BPP, IP, and PNP. We will study techniques to classify problems according to our available taxonomy. By developing a subtle pattern of reductions between classes we will suggest an (as yet unproven!) picture of how by using limited amounts of various resources, we limit our computational power. Prerequisite: 15-251.

15-462 Computer Graphics
Fall and Spring: 12 units
This course provides a comprehensive introduction to computer graphics modeling, animation, and rendering. Topics covered include basic image processing, geometric transformations, geometric modeling of curves and surfaces, animation, 3-D viewing, visibility algorithms, shading, and ray tracing. Prerequisites: (21-259 and 15-213 and 21-241) or (15-213 and 18-202).

15-463 Computational Photography
Fall: 12 units
Computational Photography is an emerging new field created by the convergence of computer graphics, computer vision and photography. Its role is to overcome the limitations of the traditional camera by using computational techniques to produce a richer, more vivid, perhaps more perceptually meaningful representation of our visual world. The aim of this advanced undergraduate course is to study ways in which samples from the real world (images and video) can be used to generate compelling computer graphics imagery. We will learn how to acquire, represent, and render scenes from digitized photographs. Several popular image-based algorithms will be presented, with an emphasis on using these techniques to build practical systems. This hands-on emphasis will be reflected in the programming assignments, in which students will have the opportunity to acquire their own images of indoor and outdoor scenes and develop the image analysis and synthesis tools needed to render and view the scenes on the computer. Prerequisites: 15-213 and (21-241 or 18-202).
Course Website: http://graphics.cs.cmu.edu/courses/15-463/
15-464 Technical Animation  
Fall: 12 units  
This course introduces techniques for computer animation such as keyframing, procedural methods, motion capture, and simulation. The course also includes a brief overview of storyboarding, scene composition, lighting and sound track generation. The second half of the course will explore current research topics in computer animation such as dynamic simulation of flexible and rigid objects, automatically generated control systems, and evolution of behaviors. The course should be appropriate for graduate students in all areas and for advanced undergraduates. Prerequisite: 15-462.

15-465 Animation Art and Technology  
Spring: 12 units  
Animation Art and Technology is an interdisciplinary course cross-listed between Art and Computer Science. Faculty and teaching assistants from computer science and art teach the class as a team. It is a project-based course in which four to five interdisciplinary teams of students produce animations. Most of the animations have a substantive technical component and the students are challenged to consider innovation with content to be equal with the technical. The class includes basic tutorials for work in Maya leading toward more advanced applications and extensions of the software such as motion capture and algorithms for animating cloth, hair, particles, and grouping behaviors. The first class will meet in CFA room 303. Prerequisite: 15-462.

15-466 Computer Game Programming  
Spring: 12 units  
The goal of the course is two-fold. The first goal is to teach students some of the higher-level techniques that are necessary to implement interesting computer games. In particular, a large emphasis will be on game AI. The class will also cover such topics as game networking, scheduling of tasks in computationally intensive games, and game design. The class will also briefly cover few selected topics such as collision detection and physically-based animation that are good to know even if one is developing games using a game engine. The second goal of the class is to get students familiar with programming games on various platforms using state-of-the-art game engines. To this end, the course will have a heavy programming focus. It will have a number of projects requiring students to apply the learned material to develop games using Unity 3D game engine. The students will develop games that run on a standalone PC, games that run on an Android platform and games that work in a multi-user mode. In addition to learning the material and learning how to use game engines, the projects will provide students with portfolios of games they have developed. Such a portfolio is important in getting a job in game industry. Prerequisite is 15-462 or contact the instructor for permission. Prerequisite: 15-462.

15-486 Artificial Neural Networks  
Intermittent: 12 units  
Artificial neural networks combine ideas from machine learning, statistics, and pattern recognition. They draw inspiration from, and provide simplified formalizations of, theories about the workings of the brain. This course offers an introduction to neural networks for computer scientists and engineers. Prerequisites are undergraduate calculus and linear algebra, and solid programming skills. An undergraduate course in artificial intelligence or machine learning would provide helpful background but is not required. The course provides hands-on experience with a variety of neural network architectures implemented in MATLAB, and an in-depth look at problems in pattern recognition and knowledge representation. Topics covered include perceptrons, the LMS learning rule, fundamentals of pattern recognition, backpropagation learning, forward and inverse models in control theory, competitive learning, self-organizing feature maps, radial basis functions, the EM algorithm, Hopfield networks, Boltzmann machines, Helmholtz machines, and general recurrent networks. Prerequisites: 15-212 and (18-202 or 21-241).

15-491 Special Topic: CMRoboBits: Creating Intelligent Robots  
Fall: 12 units  
Creating intelligent robots can be viewed as the integration of many pieces - "bits" - "RoboBits". This course will teach students such "RoboBits" for creating both single and groups of intelligent robots, with perception, cognition, and action. Students in the course work with real functional mobile robots, equipped with RGB and RGB+D cameras, different types of actuation, namely wheeled and legged robots, and computation onboard. Students learn effective vision processing algorithms for object recognition, motion modeling and planning, robot localization and planning algorithms, and multi-robot communication for shared sensing and teamwork. We will further study cognitive architectures, learning, and human-robot interaction. 15-491 is a PROJECT-BASED course, which runs in the CORAL research lab in GHC 3201. The course has three major projects with specific tasks that incrementally lead to the research and development of complete intelligent robots. The projects are decomposed in weekly tasks. The class meets twice a week, and the students have access to the lab for further work as needed. Evaluation is based on class participation, weekly progress, and project reports, presentations, and outcomes. Prerequisite: 15-122.

15-492 Special Topic: Speech Processing  
Fall: 12 units  
Speech Processing offers a practical and theoretical understanding of how human speech can be processed by computers. It covers speech recognition, speech synthesis and spoken dialog systems. The course involves practicals where the student will build working speech recognition systems, build their own synthetic voice and build a complete telephone spoken dialog system. This work will be based on existing toolkits. Details of algorithms, techniques and limitations of state of the art speech systems will also be presented. This course is designed for students wishing understand how to process real data for real applications, applying statistical and machine learning techniques as well as working with limitations in the technology. Prerequisite: 15-122.

Course Website: http://www.speech.cs.cmu.edu/15-492/

15-494 Special Topic: Cognitive Robotics  
Spring: 12 units  
Cognitive robotics is a new approach to robot programming based on high level primitives for perception and action. These primitives draw inspiration from ideas in cognitive science combined with state of the art robotics algorithms. Students will experiment with these primitives and help develop new ones using the Tekkotsu software framework on the Calliope robot, which includes a 5 degree-of-freedom arm with gripper, a Kinect camera on a pan/tilt mount, and Ubuntu Linux on a dual-core on-board netbook. Prior robotics experience is not necessary, but strong programming skills are required. Prerequisite: 15-214.

15-591 Independent Study in Computer Science  
Fall  
To be determined.

15-592 Independent Study in Computer Science  
Fall  
To be determined.

15-593 Independent Study in Computer Science  
Fall  
To be determined.

15-594 Independent Study in Computer Science  
Fall  
To be determined.

15-599 Undergraduate Thesis Research  
Fall and Spring  
Available only to students registered in the CS Senior Research Thesis Program.
15-610 Engineering Distributed Systems
Spring: 12 units
This is a master’s level course for students with strong design and implementation skills who are likely to pursue careers as software architects and lead engineers. It may be taken by well-prepared undergraduates with excellent design and implementation skills in low-level systems programming. The course assumes a high level of proficiency in all aspects of operating system design and implementation. This course will help students prepare for leadership roles in creating and evolving the complex, large-scale computer systems that society will increasingly depend on in the future. The course will teach the organizing principles of such systems, identifying a core set of versatile techniques that are applicable across many system layers. Students will acquire the knowledge base, intellectual tools, hands-on skills and modes of thought needed to build well-engineered computer systems that withstand the test of time, growth in scale, and stresses of live use. Topics covered include: caching, prefetching, damage containment, scale reduction, hints, replication, hash-based techniques, and fragmentation reduction. A substantial project component is an integral part of the course. A high level of proficiency in systems programming is expected.
Prerequisite: 15-410.
Course Website: http://www.cs.cmu.edu/~15-610

15-611 Compiler Design
Fall: 12 units
This course covers the design and implementation of compiler and run-time systems for high-level languages, and examines the interaction between language design, compiler design, and run-time organization. Topics covered include syntactic and lexical analysis, handling of user-defined types and type-checking, context analysis, code generation and optimization, and memory management and run-time organization.
Prerequisites: 15213 or 15312.
Course Website: http://symbolaris.com/course/compiler12.html

15-719 Advanced Cloud Computing
Fall: 12 units
to be determined.

15-883 Computational Models of Neural Systems
All Semesters: 12 units
This course is an in-depth study of information processing in real neural systems from a computer science perspective. We will examine several brain areas, such as the hippocampus and cerebellum, where processing is sufficiently well understood that it can be discussed in terms of specific representations and algorithms. We will focus primarily on computer models of these systems, after establishing the necessary anatomical, physiological, and psychophysical context. There will be some neuroscience tutorial lectures for those with no prior background in this area. Please refer to http://www.cs.cmu.edu/~csd-grad/courseschedulef13.html for the most recent schedule updates.
Course Website: http://www.cs.cmu.edu/afs/cs/academic/class/15883-f09/