Computer Science

Departmental Counselor:
Sharon Salveter, Ry 161B, 834-2773, salveter@cs.uchicago.edu
Student Services Representative:
Margaret Jaffey, Ry 161A, 702-6011, margaret@cs.uchicago.edu

Web: www.cs.uchicago.edu

Program of Study

The computer science program prepares students for either graduate work or employment in computer science by offering both the B.A. and B.S. degrees. Students receiving the B.A. will have sufficient breadth and depth for either graduate study or immediate employment in computer science. Recipients of the B.S. will also have substantial depth and breadth in a field outside of computer science through the completion of an approved related area.

Students in other fields of study may complete a minor in Computer Science. Information follows the description of the major.

Program Requirements

Both the B.A. and B.S. in Computer Science require fulfillment of the College's mathematical sciences requirement in general education by completing an approved two-quarter calculus sequence. The physical sciences requirement in general education must be satisfied by completing an approved two-quarter sequence in either chemistry or physics. Both B.A. and B.S. students take at least fourteen computer science courses chosen from an approved program. B.S. students also take three courses in an approved related field outside computer science.

Students pursuing a bachelor's degree in computer science should note that by judicious choice of courses from another field a supplementary field can be developed that is often in itself a solid basis for graduate or professional work in that field. Some examples are biology, biophysics, chemistry, geophysical sciences, history, linguistics, mathematics, philosophy, political science, psychology, physics, sociology, statistics, and economics.

Advanced Placement. Computer science majors may not use AP credit for computer science to meet requirements in the major. Students with AP scores of 4 or 5 on Computer Science AB receive two quarters of elective credit. NOTE: Students must forgo AP elective credit if they register for CMSC 10500, 10600, 15100, or 15200. Students who enroll in CMSC 16100 and 16200 may retain AP elective credit.

Computer science majors may use AP credit for chemistry or physics to meet their physical sciences requirement in general education or physical science components of the major. However, no credit designated simply as "physical science" (from either AP or the College's physical sciences examinations) may be used to meet general education or requirements in the computer science majors.

Approved Programs. The notion of "approval" in the program requirements allows timely response to change in the course offerings of the various departments. The computer science departmental counselor is responsible for approval of specific courses and sequences. See the departmental counselor for details on specific courses you are considering taking to meet the requirements.

Approved Computer Science Program

For the authoritative version of the Department of Computer Science requirements and course descriptions, visit *www.cs.uchicago.edu.*

There is a single approved program comprising required courses in four topical areas, plus three elective computer science courses. This is a general program in computer science and is used for either the B.A. or the B.S. degree. Upper-level or graduate courses in similar topics may be substituted for those on the list that follows, with the approval of the departmental counselor. *Students considering a computer science major are strongly advised to register for an introductory sequence in their first year.*

1. Introductory Sequence (four courses required):
   CMSC 15100 or 16100, and
   CMSC 15200 or 16200, and
   CMSC 15300, and
   CMSC 15400

2. Programming Languages and Systems Sequence (two courses required):
   Two courses from CMSC 22100, 22200, 22610, 23000, 23500, 23700

3. Algorithms and Theory Sequence (three courses required):
   CMSC 27100, and
   CMSC 27200, and
   CMSC 28000 or CMSC 28100

4. Other Sequences (one two-course sequence required):
   a. Artificial Intelligence Sequence (two courses required):
      CMSC 25000 and one of 25020, 25030, 25040, 25050, or 25100
   b. Advanced Systems Sequence (two courses required):
      CMSC 22100, 22200, 22610, 22620, 23000, 23300, 23500, or 23700 depending upon what courses the student has taken in the Programming Languages and Systems Sequence

(courses may not be used to meet both requirements)

c. Scientific Computing Sequence (two courses required): CMSC 28510 and 28520.

5. Electives (three courses required):
Three additional elective Computer Science courses numbered 20000 or above. A B.S. student with a double major in a related area may petition to have some or all of the electives be courses in the other major.

Summary of Requirements

General Education	CHEM 11101-11201/11102-11202 or equivalent*, or PHYS 12100-12200 or higher* MATH 13100-13200 or higher*
Major	14 courses in computer science from the approved program

plus the following requirements:

B.A.		B.S.	
0	no other courses required	3	approvedcourses in a related field
14		17	

* *Credit may be granted by examination.*

Grading. Computer science majors must take courses in the major for quality grades. A grade of *C-* or higher must be received in each course in the major. Any 20000-level computer science course taken as an elective beyond requirements for the major may, with consent of instructor, be taken *P/F.*

Nonmajors may take courses for either quality grades or, subject to College regulations and with consent of instructor, for *P/F* grades. A *Pass* grade is given only for work of *C-* quality or higher. Courses taken to meet general education requirements must be taken for quality grades.

Incompletes are typically given in the Department of Computer Science only to those students who have done at least 60 percent of the course's work of a passing quality and who are unable to complete all course work by the end of the quarter. Other restrictions on Incompletes are the province of individual instructors, many of whom do not permit Incompletes. Students must make arrangements in advance with instructors and obtain their written consent to receive Incompletes.

Honors. Students may earn a B.A. or B.S. degree with honors by attaining a grade of *B* or higher in all courses in the major and a grade of *B* or higher in three approved graduate computer science courses (30000-level and above). These courses may be courses taken for the major or as electives.

Students may also earn a B.A. or B.S. degree with honors by attaining the same minimum *B* grade in all courses in the major and by writing a successful bachelor's thesis as part of CMSC 29900. This thesis must be based on an approved research project that is directed by a faculty member and approved by the departmental counselor.

Recommended Sequences in Computer Science

Introductory Sequences. The kinds of computer science courses appropriate for undergraduates will vary according to each student's interests. Students interested in a general programming background are encouraged to take CMSC 15100 and 15200 (Introduction to Computer Science I, II). Students in the humanities (or others with a humanistic background) and social sciences may consider CMSC 11000-11100 (Multimedia Programming as an Interdisciplinary Art I, II). Students with a strong mathematics background should consider the CMSC 16100-16200 (Honors Introduction to Computer Science I, II). Students interested in a two-quarter introduction to the discipline should consider CMSC 10500-10600 (Fundamentals of Computer Programming I, II), or CMSC 10600-10700 (Fundamentals of Computer Programming II, III). CMSC 15100 or 16100 is recommended for students interested in further programming study. Students who are interested in Web page design and implementation should take CMSC 10100-10200.

Courses in Specific Areas of Computer Science. Students interested in artificial intelligence (AI) should take CMSC 25000 (Introduction to Artificial Intelligence) followed by any of the elective AI courses numbered 25010-25100.

Students interested in advanced programming and systems should take CMSC 22100 (Programming Languages), and CMSC 22200 (Computer Architecture). Time permitting, they should also take CMSC 22610-22620 (Implementation of Computer Languages), CMSC 23000 (Operating Systems), CMSC 23300 (Networks and Distributed Systems), CMSC 23500 (Databases), CMSC 23700 (Graphics) and such courses in advanced programming topics that may be offered.

Students interested in theoretical computer science should take CMSC 27100 (Discrete Mathematics), CMSC 27200 (Theory of Algorithms), CMSC 28000 (Formal Language and Automata) and CMSC 28100 (Introduction to Complexity Theory). Once students have completed CMSC 27100, 27200, 28000, and/or 28100, they will be qualified for most of the advanced topics courses offered at the 30000-level and above.

Students interested in numerical and scientific computing should take CMSC 28510 (Introduction to Scientific Computing) and CMSC 28520 (High-Performance Scientific Software.

The department also offers a number of special-interest courses that are detailed in the course descriptions. For information on new courses that are added on a regular basis, see the departmental counselor and visit *www.cs.uchicago.edu.*

Preparation for Graduate Study in Computer Science. Students interested in continuing their studies beyond the undergraduate level should major in computer science and take as many computer science courses as possible. The most important courses are CMSC 15100, 15200, 15300, 15400, 22100, 22200, 22610, 22620, 23000, 23300, 23500, 23700, 25000, 27100, 27200, 28000, and 28100. For more information about options for graduate study, consult the departmental counselor and the director of graduate studies.

Minor Program in Computer Science

The minor in Computer Science requires seven or eight courses, depending on which introductory computer science courses are taken. The introductory sequence of computer science courses is followed by three approved upper-level courses chosen to complement a student's major or personal interest. Courses in the minor must be taken for quality grades, with a grade of *C-* or higher in each course. Students may not use AP credit for computer science to meet requirements for the minor.

No courses in the minor can be double counted with the student's major(s) or with other minors; nor can they be counted toward general education requirements. More than half of the requirements for the minor must be met by registering for courses bearing University of Chicago course numbers. By the end of Spring Quarter of their third year, students must submit approval of their minor program from the departmental counselor on a form obtained by their College adviser.

Other programs may be approved in consultation with the departmental counselor.

Introductory Courses. Students must choose four to five courses from the following:

One course:

CMSC 10500. Fundamentals of Computer Programming I
CMSC 15100. Introduction to Computer Science I
CMSC 16100. Honors Introduction to Computer Science I

One to two courses:

CMSC 10600-10700. Fundamentals of Computer Programming II, III
CMSC 15200. Introduction to Computer Science II

CMSC 16200. Honors Introduction to Computer Science II

CMSC 15300. Foundations of Software
CMSC 15400. Introduction to Computer Systems

NOTE: Students completing the minor program who have registered for CMSC 10500, 10600, and 10700 will have met the prerequisite for CMSC 15300 and 15400.

Upper-Level Courses. Three 20000-level or above computer science courses must be approved by the departmental counselor or selected from a pre-approved three-course group below. A 20000-level course must replace each 10000-level course in the list above that was used to meet general education requirements.

Programming Languages and Systems (choose any three):
CMSC 22100. Programming Languages
CMSC 22200. Computer Architecture
CMSC 22610. Implementation of Computer Languages I
CMSC 23000. Operating Systems
CMSC 23300. Networks and Distributed Systems
CMSC 23500. Introduction to Database Systems
CMSC 23700. Introduction to Computer Graphics

Algorithms and Theory:
CMSC 27100. Discrete Mathematics
CMSC 27200. Algorithms
CMSC 28000. Introduction to Formal Languages; *or*
CMSC 28100. Introduction to Complexity Theory

Scientific Computing:
CMSC 28510. Introduction to Scientific Computing
CMSC 28520. High-Performance Scientific Software
One approved elective course

Artificial Intelligence (choose any three):
CMSC 25000. Introduction to Artificial Intelligence I
CMSC 25020. Computational Linguistics
CMSC 25030. Computational Models of Speech
CMSC 25040. Introduction to Computer Vision
CMSC 25050. Computer Vision
CMSC 25100. Introduction to Artificial Intelligence II

Faculty

Y. Amit, L. Babai, T. Dupont, P. Felzenszwalb, R. Findler, L. Fortnow, I. Foster, J. Goldsmith, L. Irakliotis, R. Kirby, C. Klivans, S. Kurtz, G.-A. Levow, D. MacQueen, K. Mulmuley, S. Nestorov, P. Niyogi, M. O'Donnell, J. Reppy, A. Rogers, S. Salveter, L. R. Scott, J. Simon,

R. Soare, R. Stevens

Courses: Computer Science (cmsc)

Graduate courses and seminars offered by the Department of Computer Science are open to College students with the consent of the instructor and the departmental counselor. See the departmental counselor for more information.

Undergraduate Courses

10100. Introduction to Programming for the World Wide Web I: HTML, Java, and CGI. *This course **does not** meet the general education requirement in the mathematical sciences.* This course teaches the basics of building and maintaining a site on the World Wide Web. We discuss Internet terminology and how the Internet and its associated technologies work. Topics include programming Web sites, hypertext markup language (HTML), Cascading Style Sheets (CSS), and Common Gateway Interface (CGI) scripts (using PERL). Students also learn how to use JavaScript to add client-side functionality. *W. Sterner, Staff. Winter.*

10200. Introduction to Programming for the World Wide Web II: HTML, CGI, and Java. *PQ: Knowledge of HTML. This course meets the general education requirement in the mathematical sciences.* This course introduces computer programming in Java. We cover the basics of object-oriented programming in Java with applications to the World Wide Web. We also discuss applets, basic graphics, and graphical user interfaces. As time permits, more advanced topics (e.g., threads, animation, Java Server Pages) are presented. *Spring.*

10500. Fundamentals of Computer Programming I: Scheme. *PQ: MATH 10600, or placement into 13100 or equivalent; or consent of departmental counselor. Previous computer experience and advanced mathematical knowledge not required. CMSC 10500 and 10600 may be taken in sequence or individually. This course meets the general education requirement in the mathematical sciences.* Both CMSC 10500 and 15100 are general introductions to computer programming. This course introduces computer programming using the functional programming language Scheme. We emphasize design, algorithm construction, and procedural/functional/data abstraction. *S. Salveter, Staff. Autumn.*

10600. Fundamentals of Computer Programming (C++) II. *PQ: MATH 10600, or placement into 13100 or equivalent; or consent of departmental counselor. CMSC 10500 and 10600 may be taken in sequence or individually. This course meets the general education requirement in the mathematical sciences.* This course is an introduction to computer programming using the object-oriented programming language C++. We emphasize design and algorithm construction. Topics include complex types, iteration, recursion, procedural/functional/data abstraction, classes, methods, inheritance, and polymorphism. *S. Salveter, Staff. Winter.*

10700. Fundamentals of Computer Programming III. *PQ: Knowledge of C++. Open only to Computer Science majors. This course meets the general education requirement in the mathematical sciences.* This course introduces data structures. Topics include linked and sequential storage allocation, linked lists, stacks, queues, trees, and graphs. *S. Salveter. Spring.*

11000-11100. Multimedia Programming as an Interdisciplinary Art I, II. (=ISHU 23500-23600) *PQ: MATH 10600, or placement into MATH 13100, or equivalent; or consent of instructor. Either course in this sequence meets the general education requirement in the mathematical sciences.* Like other classic Chicago general education courses, this sequence provides students with both practical programming skills and core ideas in computer science in interdisciplinary applications. Our ideas of the arts, the character of "images" and "texts," and the ways we form communities are being transformed by the conjunction of media and computing (e.g., QuickTime, scripting). Students program on an Apple Macintosh using an advanced programming language. This course presents techniques of problem solving, program coding, algorithm construction, and debugging using a high-level prototyping environment. We treat programs as genres of argument. *W. Sterner. Winter, 2007.*

11200. Introduction to Interactive Logic. (=ISHU 23702) *PQ: MATH 10600, or placement into 13100, or equivalent. Some experience with computers helpful. This course* ***does not*** *meet the general education requirement in the mathematical sciences.* No programming skills are assumed, but those with some programming background do projects with HyperCard, a Computer Assisted Design package, Prolog, or other software. The course continues in the same spirit as CMSC 11000-11100, but they are not prerequisites. This hands-on course presents logic as a concrete discipline that is used for understanding and creating human-computer technology in the context of science, technology, and society. We look at computer science, logic, philosophy, aesthetics, design, and the study of technology, as well as at the software packages of Tarski's World and possibly HyperProof. *W. Sterner. Spring, 2008.*

15100-15200. Introduction to Computer Science I, II. *PQ: Placement into MATH 15100 or equivalent, or consent of departmental counselor. Nonmajors may use either course in this sequence to meet the general education requirement in the mathematical sciences; Computer Science majors must use either CMSC 15100-15200 or 16100-16200 to meet requirements for the major.* This course, which is recommended for all students planning to take more advanced courses in computer science, introduces computer science using both functional (Scheme) and object-oriented (C++) programming languages. Topics include control and data abstraction, self-reference, time and space analysis, and data structures. *P. Felzenszwalb, R. Findler, R. Kirby, G.-A. Levow. Autumn, Winter, Summer.*

15300. Foundations of Software. *PQ: CMSC 15200. Required of Computer Science majors.* This course is concerned with the mathematical foundations of computer software. We introduce a number of mathematical areas used in the

modeling of programming languages and software, including prepositional and predicate logic, basic set theory, relations, and automata theory. The connection between mathematics and software is made via examples and small programming assignments. *D. MacQueen. Autumn.*

15400. Introduction to Computer Systems. *PQ: CMSC 15200. Required of Computer Science majors.* This course covers the basics of computer systems from a programmer's perspective. Topics include data re-presentation, machine language programming, exceptions, code optimization, performance measurement, memory systems, and system-level I/O. Extensive programming required. *S. Nestorov. Spring.*

16100-16200. Honors Introduction to Computer Science I, II. *PQ: Placement into MATH 16100 or equivalent, or consent of departmental counselor. Students who have taken CMSC 15100 may take CMSC 16200 with consent of instructor. Nonmajors may use either course in this sequence to meet the general education requirement in the mathematical sciences; Computer Science majors must use either CMSC 15100-15200 or 16100-16200 to meet requirements for the major.* This sequence is an honors version of CMSC 15100-15200. Topics are covered more quickly with greater depth and breadth. We emphasize studying programs as formal objects. *T. Dupont, S. Kurtz. Autumn, Winter.*

Undergraduate and Graduate Courses

Other 20000-level courses may be offered. Please see www.cs.uchicago.edu *and* timeschedules.uchicago.edu *for the most up-to-date information.*

21500. Logic and Logic Programming. (=MATH 27900) *PQ: MATH 25400 or CMSC 27700, or consent of instructor. Students are encouraged to take both CMSC 21500 and 27700. Programming knowledge not required.* Predicate logic is a precise logical system developed to formally express mathematical reasoning. Prolog is a computer language intended to implement a portion of predicate logic. This course covers both predicate logic and Prolog, which are presented to complement each other and to illustrate the principles of logic programming and automated theorem proving. Topics include syntax and semantics of propositional and predicate logic, tableaux proofs, resolution, Skolemization, Herbrand's theorem, unification, and refining resolution. Students are required to attend weekly classes and to complete programming assignments in Prolog (e.g., searching, backtracking, cut). *R. Soare. Winter, 2008.*

22001. Software Construction. *PQ: CMSC 15400.* Beyond specific domain skills, building software is a craft that requires careful design. This course teaches key software design principles in a studio setting. Each week, students present their programs to the class for a design review. Together, the class evaluates the programs for their correctness and, more importantly, their clarity and design. Students learn how to build reliable, maintainable, extensible software and how to evaluate other code for those same properties. *R. Findler. Winter, 2008.*

22100/32100. Programming Languages. *PQ: CMSC 15300.* Programming language design aims at the closest possible correspondence between the structures of a program and the task it performs. This course studies some of the structural concepts affecting programming languages: iterative and recursive control flow, data types and type checking, procedural versus functional programming, modularity and encapsulation, fundamentals of interpreting and compiling, and formal descriptions of syntax and semantics. Students write short programs in radically different languages to illuminate the variety of possible designs. *Autumn.*

22200/32200. Computer Architecture. *PQ: CMSC 15400.* This course is a survey of contemporary computer organization covering CPU design, instruction sets, control, processors, busses, ALU, memory, pipelined computers, multiprocessors, networking, and case studies. We focus on the techniques of quantitative analysis and evaluation of modern computing systems, such as the selection of appropriate benchmarks to reveal and compare the performance of alternative design choices in system design. We emphasize major component subsystems of high-performance computers: pipelining, instruction-level parallelism, memory hierarchies, input/output, and network-oriented interconnections. *A. Rogers. Spring.*

22610. Implementation of Computer Languages I. *PQ: CMSC 15300 and 15400 required; CMSC 22100 recommended. Prior experience with ML programming not required.* This course covers principles and techniques for implementing computer languages (e.g., programming languages, query languages, specification languages, domain-specific languages). Topics include lexical analysis, parsing, tree representations of programs (both parse trees and abstract syntax trees), types and type checking, interpreters, abstract machines, and run-time systems. This is a project-based course involving the implementation of a small language using Standard ML. *J. Reppy. Winter, 2007.*

22620/32620. Implementation of Computer Languages II. *PQ: CMSC 22610 required; CMSC 22100 strongly recommended.* This course is a continuation of CMSC 22610, covering compilers for general-purpose languages. Topics include compiler-immediate representations, continuation-passing style, runtime representations, code generation, code optimization, register allocation, instruction scheduling, and garbage collection. This is a project-based course in which students construct a complete, working compiler for a small language using Standard ML. *Spring, 2007.*

22630/32630. Advanced Implementation of Computer Languages. *PQ: CMSC 22100 and 22620, or equivalent.* This course explores advanced topics in the implementation of high-level programming languages that vary each year (e.g., control-flow analysis algorithms, abstract interpretation, partial evaluation, advanced optimizations, runtime system representations, garbage collection algorithms, foreign-function interfaces). Students are expected to develop both a foundational and applied understanding of these topics. *J. Reppy. Autumn, 2007.*

22800. Free Software Practicum. *PQ: UNIX programming experience and consent of instructor.* Students who are already proficient in programming are provided with the experience of working on real software and the opportunity to collaborate with distributed teams of developers. The course work consists entirely of one or more programming tasks, which must produce freely distributed code. Course work may be chosen from the bug lists and to-do lists of well-known free software projects (i.e., Gnome, KDE, Hurd, Mozilla). Students may work individually or in groups. To earn credit, the work of the student must be incorporated into the chosen project's distribution. *M. O'Donnell. Autumn, Winter, Spring.*

23000. Operating Systems. *PQ: CMSC 15400.* This course covers basic concepts of operating systems. Topics include the notion of a process, interprocess communication and synchronization, main memory allocation, segmentation, paging, linking and loading, scheduling, file systems, and security and privacy. The course is taught on a Sun workstation using UNIX. *J. Reppy. Autumn, 2006.*

23300/33300. Networks and Distributed Systems. *PQ: CMSC 23000 or consent of instructor. Basic knowledge of C, C++, and Java, as well as operating system concepts such as processes and threads.* This course focuses on the principles and techniques used in the development of networked and distributed software. Topics include programming with sockets, remote procedure calls (RPC), interprocess communication (IPC), distributed objects (i.e., CORBA and DCOM), and commonly used network protocols (e.g., TCP/IP, UDP, FTP, HTTP). In addition, data encoding, encryption, and compression algorithms are presented. This is a project-oriented course in which students are required to develop software in the UNIX programming environment. *A. Rogers, I. Foster. Winter.*

23340/33340. Grid Computing. *PQ: Substantial programming experience*. The new Open Grid Services Architecture (OGSA) defines interfaces and protocols that promise to make it far easier to construct decentralized, dynamic, large-scale systems. We explore and evaluate this technology by using it to develop a range of scalable distributed services. We use the Globus Toolkit, an open source implementation of key OGSA standards, to design and build services. We then evaluate our implementations from the perspectives of performance and programmability. *I. Foster. Winter, 2008.*

23500. Introduction to Database Systems. *PQ: CMSC 15300 required; CMSC 27100 recommended*. This course introduces database design and programming. Topics include entity-relationship and relational models, relational algebra, functional dependencies, and normal forms. We extensively cover SQL and take a brief look at data warehousing and data mining. Students design and implement e-commerce database applications using a commercial RDBMS. *S. Nestorov. Spring.*

23700. Introduction to Computer Graphics. *PQ: CMSC 15400*. This course introduces the basic concepts and techniques used in 3D computer graphics. The

focus is on real-time rendering techniques, such as those found in computer games. These include coordinate systems and transformations, the graphics pipeline, basic geometric algorithms, texture mapping, level-of detail optimizations, and shadows. Students are required to complete both written assignments and programming projects using OpenGL. *J. Reppy. Winter, 2008.*

23710/33710. Scientific Visualization. *PQ: CMSC 15200 and 23700, or consent of instructor.* This course introduces concepts and technologies for scientific visualization and advanced display environments. Topics include data models, algorithms for 2D and 3D data and information visualization, programming environments and toolkits, and interaction techniques and advanced displays (e.g., stereoscopic displays, tiled displays, virtual reality). The course is designed for students with needs for visualization of domain data, as well as for students interested in scientific visualization as a research project. *R. Stevens. Winter, 2008.*

24000. Information Theory and Coding. (=PSYC 28800/38800) *PQ: Consent of instructor.* This course is an introduction to the mathematical theory of information with emphasis on coding, especially the development of efficient codes. Topics include an introduction to coding, quantification of information and its properties, Huffman codes, arithmetic codes, L to Z, and other adaptive coding techniques and specific applications. *A. Bookstein. Winter, 2008.*

25000-25100. Introduction to Artificial Intelligence and LISP I, II. *PQ: CMSC 15300.* This course introduces the theoretical, technical, and philosophical issues of AI. We emphasize computational and mathematical modes of inquiry into the structure and function of intelligent systems. Topics include learning and inference, speech and language, vision and robotics, and search and reasoning. LISP and LISP programming are introduced. The second course looks at computational models of speech recognition. *G.-A. Levow, P. Niyogi. Winter, Spring.*

25020/35050. Computational Linguistics. *PQ: CMSC 15200 or knowledge of C++.* This course introduces the problems of computational linguistics and the techniques used to deal with them. Topics are drawn primarily from phonology, morphology, and syntax. Special topics include automatic learning of grammatical structure and the treatment of languages other than English. *This course is offered in alternate years. J. Goldsmith. Autumn.*

25030. Computational Models of Speech. *PQ: CMSC 25000.* This course covers computational models of speech: acoustic-phonetic recognition, speech processing, and Fourier analysis. *P. Niyogi. Spring, 2008.*

25040/35040. Introduction to Computer Vision. *PQ: CMSC 25000.* This course introduces some of the fundamental concepts in computer vision. We cover selected topics in low-level image analysis, perceptual grouping, 3D reconstruction, motion estimation, and object recognition. Our approach is

to focus on mathematical models and efficient algorithms. *P. Felzenszwalb. Autumn.*

25050/35500. Computer Vision. *PQ: CMSC 25000.* This course covers methods and models for computer vision. Topics include segmentation and feature detection. *Y. Amit. Winter.*

27100. Discrete Mathematics. *PQ: Placement into MATH 15100 or equivalent, or consent of departmental counselor. This is a directed course in mathematical topics and techniques needed by students taking Algorithms (CMSC 27200). It is also a prerequisite to several other courses, including Honors Combinatorics and Probability (CMSC 27400/MATH 28400).* This course emphasizes mathematical discovery and rigorous proof, which are illustrated on a refreshing variety of accessible and useful topics. Basic counting is a recurring theme and provides the most important source for sequences, which is another recurring theme. Further topics include proof by induction; recurrences and Fibonacci numbers; graph theory and trees; number theory, congruences, and Fermat's little theorem; counting, factorials, and binomial coefficients; combinatorial probability; random variables, expected value, and variance; and limits of sequences, asymtotic equality, and rates of growth. *C. Klivans. Autumn.*

27200. Theory of Algorithms. *PQ: CMSC 27100 or consent of instructor.* This course covers design and analysis of efficient algorithms, with emphasis on ideas rather than on implementation. Algorithmic questions include sorting and searching, discrete optimization, algorithmic graph theory, algorithmic number theory, and cryptography. Design techniques include "divide-and-conquer" methods, dynamic programming, greedy algorithms, and graph search, as well as the design of efficient data structures. Methods of algorithm analysis include asymptotic notation, evaluation of recurrent inequalities, the concepts of polynomial-time algorithms, and NP-completeness. *P. Felzenszwalb. Winter.*

27400. Honors Combinatorics and Probability. (=MATH 28400) *PQ: MATH 25000 or 25400, or CMSC 27100, or consent of instructor. Experience with mathematical proofs.* Methods of enumeration, construction, and proof of existence of discrete structures are discussed in conjunction with the basic concepts of probability theory over a finite sample space. Enumeration techniques are applied to the calculation of probabilities, and, conversely, probabilistic arguments are used in the analysis of combinatorial structures. Other topics include basic counting, linear recurrences, generating functions, Latin squares, finite projective planes, graph theory, Ramsey theory, coloring graphs and set systems, random variables, independence, expected value, standard deviation, and Chebyshev's and Chernoff's inequalities. *This course is offered in alternate years. L. Babai. Spring, 2008.*

27500. Graph Theory. *PQ: CMSC 15300.* This course covers the basics of the theory of finite graphs. Topics include shortest paths, spanning trees, counting techniques, matchings, Hamiltonian cycles, chromatic number, extremal graph

theory, Turan's theorem, planarity, Menger's theorem, the max-flow/min-cut theorem, Ramsey theory, directed graphs, strongly connected components, directed acyclic graphs, and tournaments. Techniques studied include the probabilistic method. *J. Simon. Winter.*

27700. Mathematical Logic I. (=MATH 27700) *PQ: MATH 25400.* For course description, see Mathematics. *J. Mileti. Autumn.*

27800. Mathematical Logic II. (=MATH 27800) *PQ: MATH 27700 or equivalent.* For course description, see Mathematics. *J. Mileti. Winter.*

28000. Introduction to Formal Languages. (=MATH 28000) *PQ: MATH 25000 or 25500, and experience with mathematical proofs.* This course is a basic introduction to computability theory and formal languages. Topics include automata theory, regular languages, context-free languages, and Turing machines. *S. Kurtz. Autumn.*

28100. Introduction to Complexity Theory. (=MATH 28100) *PQ: CMSC 27100, or MATH 25000 or 25500; and experience with mathematical proofs.* Computability topics are discussed (e.g., the s-m-n theorem and the recursion theorem, resource-bounded computation). This course introduces complexity theory. Relationships between space and time, determinism and nondeterminism, NP-completeness, and the P versus NP question are investigated. *L. Fortnow. Spring.*

28501. Topics in Scientific Computing. (=GEOS 21000) *PQ: Consent of instructor required; programming experience not required.* This course introduces the mathematical background necessary for understanding the behavior of time-evolving systems, the numerical techniques needed for simulating such systems, and the programming techniques necessary for flexible and accurate implementation of the numerical methods. Topics include chaos in the logistic map, Python for scientific simulation, ordinary differential equations, elementary solution techniques, numerical methods for evaluating integrals, numerical methods for ordinary differential equations, and asymptotic approximations and their various uses. *R. Pierrehumbert. Autumn, 2007.*

28510. Introduction to Scientific Computing. *PQ: A year of calculus (MATH 15300 or higher), a quarter of linear algebra (MATH 25000 or higher), and CMSC 10600 or higher; or consent of instructor.* Basic processes of numerical computation are examined from both an experimental and theoretical point of view. The course deals with numerical linear algebra, approximation of functions, approximate integration and differentiation, Fourier transformation, solution of nonlinear equations, and the approximate solution of initial value problems for ordinary differential equations. The course concentrates on a few widely used methods in each area covered. *T. Dupont. Autumn, 2007.*

28520. High-Performance Scientific Software. *PQ: CMSC 28510 or consent of instructor.* This course revisits many numerical algorithms studied in CMSC

28510 from a software performance perspective. We examine issues of single-processor performance and code reusability, including object orientation and interfacing with standard packages (e.g., BLAS, LAPACK). The linear algebraic tools discussed early in the class are used to address problems in interpolation, integration, nonlinear equations, and differential equations. *R. Kirby. Winter, 2008.*

29500. Digital Sound Modeling. *PQ: Knowledge of computer programming or consent of instructor.* Students taking this course study how the basic structure of sound perception affects the useful ways of representing sound in digital computations. We focus on basic synthesis techniques, rather than on signal analysis or on special applications of synthesis (e.g., music, speech). Course work is divided among intuitive mathematical studies, listening exercises, and a cooperative project using synthesis software to simulate an orchestral instrument. *M. O'Donnell. Winter.*

29700. Reading and Research in Computer Science. *PQ: Consent of instructor and approval of departmental counselor. Open to both Computer Science majors and nonmajors. Students are required to submit the College Reading and Research Course Form.* Students do reading and research in an area of computer science under the guidance of a faculty member. A written report is typically required. *Summer, Autumn, Winter, Spring.*

29900. Bachelor's Thesis. *PQ: Open to fourth-year students who are candidates for honors in computer science. Consent of instructor and departmental counselor. Students are required to submit the College Reading and Research Course Form. Summer, Autumn, Winter, Spring.*

Graduate Courses

College students may register for graduate courses with consent of departmental counselor. For details, visit www.cs.uchicago.edu.

31100. Big Ideas in Computer Science. *PQ: Consent of instructor.* This course introduces many of the important concepts in the broad area of computer science. Each week a different professor gives a three-lecture sequence on a big idea in their field of specialty. Previous ideas have included undecidability, randomness, cryptography, stability of numerical algorithms, structural operational semantics, software engineering, and the Internet. *M. O'Donnell. Autumn.*

31900. Lambda Calculus. *PQ: Knowledge of algebra or equivalent level of math.* This course covers the crucial properties of the lambda calculus and its variant the combinator calculus, emphasizing the deep connections between various areas of logic and computation that arise from the ability to interpret a lambda term

equally naturally as a program of type A to B and as a proof that A implies B. *M. O'Donnell. Autumn, 2007.*

32001. Topics in Programming Languages. *PQ: Consent of instructor.* This course covers a selection of advanced topics in programming languages. *Autumn, Winter, Spring.*

32201. Topics in Computer Architecture. *PQ: Consent of instructor.* This course covers a selection of advanced topics in computer architecture, with "Parallel Computer Architecture" the topic for Autumn Quarter. *Autumn, Winter, Spring.*

33501. Topics in Databases. *PQ: Consent of instructor.* This course covers a selection of advanced topics in database systems. *Autumn, Winter, Spring.*

33510. Data Mining. *PQ: CMSC 15200 or 11700, and 27100; or consent of instructor.* In addition to being an emerging field at the intersection of machine learning, statistics, and databases, the broad definition of data mining is finding novel and interesting patterns in large amounts of data. In this research-oriented course, we survey data-mining techniques and applications, emphasizing the database perspective. Themes include association rules, Web search and mining, clustering, and sequence and stream mining. The course involves an independent data-mining project. *S. Nestorov. Winter.*

33600. Type Systems for Programming Languages. *PQ: CMSC 22100 recommended.* This course covers the basic ideas of type systems, their formal properties, their role in programming language design, and their implementation. Exercises involving design and implementation explore the various options and issues. *D. MacQueen. Winter, 2008.*

34000. Scientific Parallel Computing. *PQ: Consent of instructor required; experience in scientific computing helpful.* We study the use of multiple processors cooperating to solve a common task, as well as related issues in computer architecture, performance analysis, prediction and measurement, programming languages, and algorithms for large-scale computation. Programming at least one parallel computer is required. Possibilities include one of the clusters of workstations connected by high-speed networks on campus. We focus on state-of-the-art parallel algorithms for scientific computing. Topics are based on interest. General principles of parallel computing are emphasized. *L. R. Scott. Autumn.*

34200. Numerical Hydrodynamics. *PQ: Ability to program, and familiarity with elementary numerical methods and modeling physical systems by systems of differential equations.* This course covers numerical methods for the solution of fluid flow problems. We also make a theoretical evaluation of the methods and experimental study based on the opinionated book *Fundamentals of Computational Fluid Dynamics* by Patrick J. Roache. *T. Dupont. Spring.*

34710. Wireless Sensor Networks. *PQ: Graduate-level understanding of Unix/Linux operating systems, networking, computer architecture, and programming.* This

course introduces the concepts and technologies for building embedded systems and wireless sensors nets by focusing on four areas: low-power hardware, wireless networking, embedded operating systems, and sensors. Two assignments provide hands-on experience by deploying small wireless sensor motes running TinyOS to form an ad-hoc peer-to-peer network that can collect environmental data and forward it back to an 802.11b-equiped embedded Linux module. Students also read and summarize papers, participate in classroom discussions, and work on a team research project. *R. Stevens. Winter, 2008.*

35000. Introduction to Artificial Intelligence. *PQ: CMSC 25000.* This course is an introduction to the theoretical, technical, and philosophical aspects of Artificial Intelligence. The emphasis is on computational and mathematical modes of inquiry into the structure and function of intelligent systems. Topics include learning and inference, speech and language, vision and robotics, and reasoning and search. *P. Niyogi. Winter.*

35100. Natural Language Processing. *PQ: CMSC 25000/35000 or consent of instructor.* This course is an introduction to the theory and practice of natural language processing, with applications to both text and speech. Topics include regular expressions, finite state automata, morphology, part of speech tagging, context free grammars, parsing, semantics, discourse, and dialogue. Symbolic and probabilistic models are presented. Techniques for automatic acquisition of linguistic knowledge are emphasized. *G.-A. Levow. Winter, 2008.*

35400. Machine Learning. *PQ: CMSC 25000/35000 or consent of instructor.* This course is an introduction to the theory and practice of machine learning that emphasizes statistical approaches to the problem. Topics include pattern recognition, empirical risk minimization and the Vapnik Chervonenkis theory, neural networks, decision trees, genetic algorithms, unsupervised learning, and multiple classifiers. *P. Niyogi. Spring.*

35500. Computer Vision. *PQ: Consent of instructor.* This course covers deformable models for detecting objects in images. Topics include one-dimensional models to identify object contours and boundaries; two-dimensional models for image matching; and sparse models for efficient detection of objects in complex scenes. Mathematical tools needed to define the models and associated algorithms are developed. Applications include detecting contours in medical images, matching brains, and detecting faces in images. Neural network implementations of some of the algorithms are presented, and connections to the functions of the biological visual system are discussed. *Y. Amit. Winter.*

35900. Topics in Artificial Intelligence. *PQ: Consent of instructor.* Current topics in artificial intelligence are covered in this course. Topics in 2006-07 include "Discourse" in Autumn and "Topics in Machine Learning" in Winter. *G.-A. Levow, Autumn; Y. Altun, Winter; Staff, Spring.*

36500. Algorithms in Finite Groups. (=MATH 37500) *PQ: Linear algebra, finite fields, and a first course in group theory (Jordan-Holder and Sylow theorems). Prior knowledge of algorithms not required; there will be no programming problems.* We consider the asymptotic complexity of some of the basic problems of computational group theory. The course demonstrates the relevance of a mix of mathematical techniques, ranging from combinatorial ideas, the elements of probability theory, and elementary group theory, to the theories of rapidly mixing Markov chains, applications of simply stated consequences of the Classification of Finite Simple Groups (CFSG), and, occasionally, detailed information about finite simple groups. *L. Babai. Spring, 2008.*

37000. Algorithms. *PQ: CMSC 27200 or consent of instructor.* The focus of this course is the analysis and design of efficient algorithms, with emphasis on ideas rather than on implementation. Algorithmic questions include sorting and searching, discrete optimization, algorithmic graph theory, algorithmic number theory, and cryptography. Design techniques include "divide-and-conquer" methods, dynamic programming, greedy algorithms, and graph search, as well as the design of efficient data structures. Methods of algorithm analysis include asymptotic notation, evaluation of recurrent inequalities, the concepts of polynomial-time algorithms, and NP-completeness. *L. Babai. Winter.*

37100. Topics in Algorithms. *PQ: CMSC 27200 or consent of instructor.* This course discusses current topics in algorithms. *Autumn, Winter, Spring.*

37110. Discrete Mathematics. *PQ: Consent of instructor.* This course emphasizes mathematical discovery and rigorous proof, which are illustrated on a refreshing variety of accessible and useful topics. Basic counting is a recurring theme and provides the most important source for sequences, which is another recurring theme. Further topics include proof by induction; recurrences and Fibonacci numbers; graph theory and trees; number theory, congruences, and Fermat's little theorem; counting, factorials, and binomial coefficients; combinatorial probability; random variables, expected value, and variance; and limits of sequences, asymtotic equality, and rates of growth. *L. Babai. Autumn.*

37200. Combinatorics. *PQ: Linear algebra, basic combinatorics, or consent of instructor.* Methods of enumeration, construction, and proof of existence of discrete structures are discussed. The course emphasizes applications of linear algebra, number theory, and the probabilistic method to combinatorics. Applications to the theory of computing are indicated, and open problems are discussed. *This course is offered in alternate years. L. Babai. Spring, 2008.*

37300. Parallel Algorithms. *PQ: CMSC 27200 or consent of instructor.* This course discusses models of parallel computation: shared memory and networks. Topics include basic algorithmic techniques (e.g., parallel prefix computation, list ranking, tree-contraction routing problems, complexity classes, completeness results) and randomized parallel algorithms. *J. Simon. Winter, 2008.*

37400. Constructive Combinatorics. *PQ: Advanced knowledge of mathematics and consent of instructor.* This course covers constructive combinatorial techniques in areas such as enumerative combinatorics, invariant theory, and representation theory of symmetric groups. Constructive techniques refer to techniques that have algorithmic flavor, such as those that are against purely existential techniques based on counting. *K. Mulmuley. Spring, 2008.*

37701. Topics in Bioinformatics. *PQ: Consent of instructor.* Current topics in bioinformatics are covered in this course. *Autumn, Winter, Spring.*

37720. Computational Systems Biology. *PQ: Consent of instructor.* This course introduces concepts of systems biology. We also discuss computational methods for analysis, reconstruction, visualization, modeling, and simulation of complex cellular networks (e.g., biochemical pathways for metabolism, regulation, and signaling). Students explore systems of their own choosing and participate in developing algorithms and tools for comparative genomic analysis, metabolic pathway construction, stoichiometeric analysis, flux analysis, metabolic modeling, and cell simulation. We also focus on understanding the computer science challenges in the engineering of prokaryotic organisms. *R. Stevens. Autumn, 2007.*

37800. Numerical Computation for Statistics. (=STAT 30700) *PQ: Elementary programming experience or consent of instructor.* For course description, see Statistics. *R. Kirby. Autumn.*

38000-38100. Computability Theory I, II. (=MATH 30200-30300) *PQ: MATH 25500 or consent of instructor.* CMSC 38000 is concerned with recursive (computable) functions and sets generated by an algorithm (recursively enumerable sets). Topics include various mathematical models for computations (e.g., Turing machines and Kleene schemata, enumeration and s-m-n theorems, the recursion theorem, classification of unsolvable problems, priority methods for the construction of recursively enumerable sets and degrees). CMSC 38100 treats classification of sets by the degree of information they encode, algebraic structure and degrees of recursively enumerable sets, advanced priority methods, and generalized recursion theory. *This course is offered in alternate years. R. Soare. Winter, Spring. Offered 2007-08; not offered 2006-07.*

38300. Numerical Solutions to Partial Differential Equations. (=MATH 38300) *PQ: Consent of instructor.* This course covers the basic mathematical theory behind numerical solution of partial differential equations. We investigate the convergence properties of finite element, finite difference and other discretization methods for solving partial differential equations, introducing Sobolev spaces and polynomial approximation theory. Special emphasis is on error estimators, adaptivity, and optimal-order solvers for linear systems arising from PDEs. Special topics include PDEs of fluid mechanics, max-norm error estimates, and Bananch-space operator-interpolation techniques. *T. Dupont. Spring, 2008.*

38500. Computability and Complexity Theory. (=MATH 30500) *PQ: Consent of instructor.* Part one consists of models for defining computable functions: primitive recursive functions, (general) recursive functions, and Turing machines; the Church-Turing Thesis; unsolvable problems; diagonalization; and properties of computably enumerable sets. Part two deals with Kolmogorov (resource bounded) complexity: the quantity of information in individual objects. Part three covers functions computable with time and space bounds of the Turing machine: polynomial time computability, the classes P and NP, NP-complete problems, polynomial time hierarchy, and P-space complete problems. *L. Fortnow. Autumn, 2006.*

38600. Complexity Theory A. *PQ: Consent of instructor.* Topics are covered in computational complexity theory with an emphasis on machine-based complexity classes. *J. Simon. Autumn.*

38700. Complexity Theory B. *PQ: Consent of instructor.* Topics are covered in computational complexity theory with an emphasis on combinatorial problems in complexity. *L. Fortnow. Spring, 2008.*

39000. Computational Geometry. *PQ: Consent of instructor.* This course is a seminar on topics in computational geometry. *K. Mulmuley. Autumn.*

39600. Topics in Theoretical Computer Science. *PQ: Consent of instructor.* This course is a seminar on current research in theoretical computer science. *Autumn, Winter, Spring.*