The major in computer science is designed to introduce students to the two fundamental questions of the discipline: What computational tasks is a computer capable of doing? How can we design, analyze, and implement efficient algorithms to solve large, complex problems? Thus, the discipline requires thinking in both abstract and concrete terms and the major provides an opportunity for students to develop the analytical skills necessary for efficient algorithm design as well as the practical skills necessary for the implementation of those algorithms. The range of problems that can be attacked using the techniques of computer science spans many disciplines, and computer scientists often become proficient in other areas. Examples of problems that students can study in the department include cryptography and network security, geographic information systems, robotics, artificial intelligence in computer games, and planning under uncertainty. The computer science major can serve as preparation for graduate study in computer science as well as careers in teaching, research, and industry (such as financial services and Internet-related businesses).

Requirements for the Major in Computer Science
The major consists of eight computer science courses and three mathematics courses. The computer science portion of the major consists of an introductory course, Computer Science 101; four intermediate “core” courses (Computer Science 210, 231, 270, and 289); two 300-level elective courses; and a third elective that may be satisfied by any remaining course numbered 260 or above, or an independent study. The mathematics portion of the major consists of Mathematics 161, or the equivalent; Mathematics 200; and another mathematics course numbered 165 or higher. Prospective majors should take Computer Science 210 and Mathematics 200 as soon as possible after Computer Science 101, since one or both of these courses are prerequisites for all other computer science courses.

Students, particularly those who intend to do graduate work in computer science or a related field, are encouraged to collaborate with faculty on research projects through independent studies, honors projects, and fellowship-funded summer research.

Computer science shares interests with a number of other disciplines, e.g. probability and statistics in mathematics, logic in philosophy, and cognition in psychology. In addition, computers are increasingly being used as a tool in other disciplines, including the social sciences and the humanities as well as the natural sciences. The department encourages students to explore these relationships; courses that may be of particular interest include Mathematics 165, 201, 204 (formerly Mathematics 174), 225, and 265; Philosophy 210, 223, and 233; Psychology 216 and 270; Music 218; and Visual Arts 255.

Requirements for the Minor in Computer Science
The minor consists of five courses: a 100-level computer science course or the equivalent, Computer Science 210, and any three additional computer science courses at the 200 level or above.
Interdisciplinary Major
The department participates in an interdisciplinary major program in computer science and mathematics. See page 204.

Fulfilling Requirements
To fulfill the major or minor requirements, or to serve as a prerequisite for another computer science course, a grade of C- or better must be earned in a course. Courses taken with the Credit/D/Fail grading option may not be used to fulfill major or minor requirements.

Introductory Courses
[50a - MCSR. Computing: Tools and Issues.]
101a - MCSR. Introduction to Computer Science. Every semester. The Department.
What is computer science, what are its applications in other disciplines, and what is its impact in society? A step-by-step introduction to the art of problem solving using the computer and the Java language. Provides a broad introduction to computer science and programming through real-life applications. Weekly labs provide experiments with the concepts presented in class. Assumes no prior knowledge of computers or programming.

Intermediate and Advanced Courses
210a - MCSR. Data Structures. Every semester. Laura Toma.
Solving complex algorithmic problems requires the use of appropriate data structures such as stacks, priority queues, search trees, dictionaries, hash tables, and graphs. It also requires the ability to measure the efficiency of operations such as sorting and searching in order to make effective choices among alternative solutions. Offers a study of data structures, their efficiency, and their use in solving computational problems. Laboratory exercises provide an opportunity to design and implement these structures.
Prerequisite: Computer Science 101 or permission of the instructor. Students interested in taking Computer Science 210 are required to pass the computer science placement examination before class starts.

231a - MCSR. Algorithms. Every fall. Laura Toma.
An introductory course on the design and analysis of algorithms building on concepts from Computer Science 210. Introduces a number of basic algorithms for a variety of problems such as searching, sorting, selection, and graph problems (e.g. spanning trees and shortest paths). Discusses analysis techniques, such as recurrences and amortization, as well as algorithm design paradigms such as divide-and-conquer, dynamic programming, and greedy algorithms.
Prerequisite: Computer Science 210 and Mathematics 200, or permission of the instructor.

[250a - MCSR. Principles of Programming Languages.]
[260a - MCSR. Software Design.]
Explores the principles and techniques involved in programming computers to do tasks that would require intelligence if people did them. State-space and heuristic search techniques, logic and other knowledge representations, reinforcement learning, neural networks, and other approaches are applied to a variety of problems with an emphasis on agent-based approaches.
Prerequisite: Computer Science 210 and Mathematics 200, or permission of the instructor.
Projects in Computer Science.


Studies the nature of computation and examines the principles that determine what computational capabilities are required to solve particular classes of problems. Topics include an introduction to the connections between language theory and models of computation, and a study of unsolvable problems.

Prerequisite: Mathematics 200 or permission of the instructor.


Robotics is a challenging discipline that encourages students to apply theoretical ideas from a number of different areas—artificial intelligence, cognitive science, operations research—in pursuit of an exciting, practical application: programming robots to do useful tasks. Two of the biggest challenges are building effective models of the world using inaccurate and limited sensors, and using such models for efficient robotic planning and control. Addresses these problems from both a theoretical perspective (computational complexity and algorithm development) and a practical perspective (systems and human/robot interaction) through multiple programming projects involving simulated and actual robots.

Prerequisite: Computer Science 210 and Mathematics 200, or permission of the instructor.


The smooth functioning of our society increasingly depends on the flow of information through computer networks. Problems of privacy and authenticity of information have become extremely important, and cryptography is an essential tool in addressing these issues. An introduction to modern cryptography, covering topics such as block ciphers, modes of operation, private-key encryption, hash functions, digital signatures, public-key encryption, RSA, the discrete logarithm problem, public-key infrastructure, key distribution, and various applications. Emphasizes a rigorous mathematical approach including formal definitions of security goals and proofs of protocol security, and identification of weaknesses in designs.

Prerequisite: Computer Science 210 and Mathematics 200, or permission of the instructor.


In many disciplines the data being collected have spatial coordinates. Analysis of spatial data is an active area of research in computer science, with applications in areas like computer-aided design (CAD), data warehousing, network routing, and geographic information systems (GIS). Presents algorithms and data structures for problems involving spatial data, covering both their theory and their practical efficiency and scalability to large datasets. Topics include spatial database design; computational geometry, covering algorithms for computing convex hulls, Delaunay triangulations and Voronoi diagrams; line segment intersection and spatial join; data structures for orthogonal range searching; nearest-neighbor queries and window queries; techniques for dynamization of spatial data structures; clustering techniques and external memory algorithms.

Prerequisite: Computer Science 210 and Mathematics 200, or permission of the instructor.


Geographic information systems (GIS) handle geographical data such as boundaries of countries; course of rivers; height of mountains; and location of cities, roads, railways, and power lines. GIS can help determine the closest public hospital, find areas susceptible to
flooding or erosion, track the position of a car on a map, or find the shortest route from one location to another. Because GIS deal with large datasets, making it important to process data efficiently, they provide a rich source of problems in computer science. Topics covered include data representation, triangulation, range searching, point location, map overlay, meshes and quadtrees, terrain simplification, and visualization.

Prerequisite: **Computer Science 210** and **Mathematics 200**, or permission of the instructor.

Advances in computer science, psychology, and neuroscience have shown that humans process information in ways that are very different from those used by computers. Explores the architecture and mechanisms that the human brain uses to process information. In many cases, these mechanisms are contrasted with their counterparts in traditional computer design. A central focus is to discern when the human cognitive architecture works well, when it performs poorly, and why. Conceptually oriented, drawing ideas from computer science, psychology, and neuroscience. No programming experience necessary.

Prerequisite: One of the following: **Computer Science 231** or **250**, **Biology 214** or **253**, or **Psychology 270**, or permission of the instructor.

**360a. Computer and Network Security.** Fall 2010. **Adriana Palacio.**
Covers the fundamental concepts and techniques used to ensure secure computing and communication. Topics include cryptographic protocols, code security and exploitation (buffer overflows, race conditions, SQL injection, etc.), access control and authentication, covert channels, protocol attacks, firewalls, intrusion detection/prevention, viruses/worms and bots, spyware and phishing, denial-of-service, privacy/anonymity, and computer forensics. Provides an appreciation of the fundamental challenges in designing and implementing secure systems as well as an understanding of the base technologies and threats in today’s interconnected environment.

Prerequisite: **Computer Science 210** or permission of the instructor.

**375a. Optimization and Uncertainty.** Spring 2010. **Stephen Majercik.**
Optimization problems and the need to cope with uncertainty arise frequently in the real world. A numeric framework, rather than the symbolic one of traditional artificial intelligence, is useful for expressing such problems. In addition to providing a way of dealing with uncertainty, this approach sometimes permits performance guarantees for algorithms. Topics include constraint satisfaction, systematic and non-systematic search techniques, probabilistic inference and planning, and population-based optimization techniques (e.g., genetic algorithms and ant colony optimization).

Prerequisite: **Computer Science 210** and **Mathematics 200**, or permission of the instructor.

**380a. Artificial Intelligence and Computer Games.** Spring 2009. **Stephen Majercik.**
Computer games are becoming an increasingly utilized test-bed for the development of new techniques in certain areas of artificial intelligence (AI) research (knowledge representation; search; planning, reasoning, and learning under uncertainty). At the same time, AI techniques are becoming increasingly necessary in commercial computer games to provide interesting and realistic synthetic characters. Explores that symbiosis by studying a subset of relevant AI techniques, using those techniques to create AI-endowed characters, and testing the characters in actual computer games.

Prerequisite: **Computer Science 210** or permission of the instructor.

**401a–404a. Advanced Independent Study and Honors in Computer Science.** The Department.