COMPUTER SCIENCE

Director of undergraduate studies: Y. Richard Yang, AKW 208A, 432-6400, yang.r.yang@yale.edu; cpsc.yale.edu

The Department of Computer Science offers both B.S. and B.A. degree programs, as well as three combined majors in cooperation with other departments: Electrical Engineering and Computer Science, Computer Science and Mathematics, and Computer Science and Psychology. Each major program not only provides a solid technical education but also allows students either to take a broad range of courses in other disciplines or to complete the requirements of a second major.

The Computer Science and combined major programs share a common core of five computer science courses. The first is CPSC 201, a survey that demonstrates the breadth and depth of the field to students who have taken the equivalent of an introductory programming course. The remaining core courses cover discrete mathematics, data structures, systems programming and computer architecture, and algorithm analysis and design. Together these courses include the material that every major should know.

The core courses are supplemented by electives (and, for the combined majors, core courses in the other discipline) that offer great flexibility in tailoring a program to each student’s interests. The capstone is the senior project, through which students experience the challenges and rewards of original research under the guidance of a faculty mentor.

Prospective majors are encouraged to discuss their programs with the director of undergraduate studies as early as possible.

INTRODUCTORY COURSES

The department offers a broad range of introductory courses to meet the needs of students with varying backgrounds and interests. Except for CPSC 200 and CPSC 201, none assumes previous knowledge of computers.

1. CPSC 100, taught jointly with Harvard University, teaches students majoring in any subject area how to program a computer and solve problems. No prior experience is required.
2. CPSC 112 teaches students majoring in any subject area how to program a computer and solve problems using the language Java. Students with previous programming experience should consider taking CPSC 201 instead.
3. CPSC 134 provides an introduction to computer music, including musical representations for computing, automated music analysis and composition, interactive systems, and virtual instrument design.
4. CPSC 150 explores how some of the key ideas in computer science have affected philosophy of mind, cognitivism, connectionism, and related areas. This humanities-style course requires a significant amount of reading and writing a paper, and satisfies the writing and the humanities and arts distributional requirements.
5. CPSC 151 studies the history of the graphical user interface in an attempt to guess its future. This course also satisfies the writing distributional requirement.
6. CPSC 183 explores the myriad ways that law and technology intersect, with a special focus on the role of cyberspace. This course satisfies the social science distributional requirement.
7. CPSC 200, intended as a survey course for non-majors, focuses on practical applications of computing technology while examining topics including computer hardware, computer software, and related issues such as security and software engineering.
8. CPSC 201 surveys the field of computer science, including systems (computers and their languages) and theory (algorithms, complexity, and computability). Students with sufficient programming experience may elect CPSC 201 without taking CPSC 112. (These courses meet at the same time so that students are easily able to change levels if necessary).
9. CPSC 202 presents the formal methods of reasoning and the concepts of discrete mathematics and linear algebra used in computer science and related disciplines.

REQUIREMENTS OF THE MAJOR

The B.S. and the B.A. degree programs have the same required five core courses: CPSC 201; CPSC 202 or MATH 244; CPSC 223; CPSC 323; and CPSC 365 or 366. CPSC 480 and 490 may not be counted toward these core courses.

B.S. degree program The B.S. degree program requires a total of twelve courses, six additional intermediate or advanced courses in Computer Science and the senior requirement.

B.A. degree program The B.A. degree program requires a total of ten courses, four additional intermediate or advanced course in Computer Science and the senior requirement.

Combined B.S./M.S. degree Exceptionally able and well-prepared students may complete a course of study leading to the simultaneous award of the B.S. and M.S. degrees after eight terms of enrollment. Eligibility requirements are described under “Simultaneous Award of the Bachelor’s and Master’s Degrees” in Section K, Special Arrangements, in the Academic Regulations. Specific requirements for the combined degree in Computer Science are as follows:
1. Candidates must satisfy the Yale College requirements for the B.S. degree in Computer Science.

2. In fulfilling these requirements, students must complete eight graduate courses from the approved list, up to two of which may, with the permission of the director of undergraduate studies and the director of graduate studies, also be applied toward completion of the B.S. degree. At most, one of these eight courses may be CPSC 690, 691, or 692.

3. At the end of their fifth term of enrollment students must have achieved at least three-fourths A or A– grades in all of their course credits directly relating to the major.

**Credit/D/Fail** Courses taken Credit/D/Fail may not be counted toward the major. All courses in the major must be taken for a letter grade.

**Roadmap** See visual roadmap of the requirements.

**SENIOR REQUIREMENT**

In the senior year students must take CPSC 490, an independent project course in which students select an adviser to guide them in research in a subfield of computer science. With permission of the director of undergraduate studies, students may enroll in 490 more than once or before their senior year.

**ADVISING**

All Computer Science majors in the sophomore, junior, and senior years should have their programs approved by the director of undergraduate studies. Students majoring in Computer Science are advised to complete CPSC 201 and 223 by the end of the sophomore year.

**Electives** The Computer Science department encourages interdisciplinary study in which computer science plays a major role. Advanced courses in other departments that involve concepts from computer science and are relevant to an individual program may, with permission of the director of undergraduate studies, be counted toward the requirements.

Students considering graduate study in computer science are advised to take CPSC 421 and 422, as well as courses covering the breadth of computer science, including programming languages and systems, artificial intelligence, scientific computing, and theoretical computer science.

Students interested in using computers to solve scientific and engineering problems are advised to take CPSC 440 as well as computational courses offered in Applied Mathematics and in Engineering and Applied Science.

The core mathematical background necessary to complete the Computer Science major is provided in CPSC 202. However, many advanced courses in graphics, computer vision, neural networks, and numerical analysis assume additional knowledge of linear algebra and calculus. Students who plan to take such courses as electives and who are unsure whether they have the appropriate mathematical background are encouraged to take MATH 222 or 225 and MATH 120.

**Typical programs** For students who already know how to program, typical B.S. programs starting in the first and sophomore years are indicated below. For typical B.A. programs, two of the electives would be omitted.

<table>
<thead>
<tr>
<th>First-Year</th>
<th>Sophomore</th>
<th>Junior</th>
<th>Senior</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPSC 201</td>
<td>CPSC 202</td>
<td>Two electives</td>
<td>CPSC 490</td>
</tr>
<tr>
<td>CPSC 223</td>
<td>CPSC 365 or 366</td>
<td>Two electives</td>
<td>One elective</td>
</tr>
<tr>
<td></td>
<td>One elective</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sophomore</td>
<td>Junior</td>
<td>Senior</td>
<td></td>
</tr>
<tr>
<td>CPSC 223</td>
<td>CPSC 365 or 366</td>
<td>Two electives</td>
<td></td>
</tr>
<tr>
<td>CPSC 201</td>
<td>CPSC 323</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPSC 202</td>
<td>One elective</td>
<td>Two electives</td>
<td></td>
</tr>
</tbody>
</table>

**REQUIREMENTS OF THE MAJOR**

**Prerequisites** None

**Number of courses** B.S. — 12 term courses taken for letter grades (incl senior project); B.A. — 10 term courses taken for letter grades (incl senior project)

**Specific courses required** B.S. and B.A. — CPSC 201; CPSC 202 or MATH 244; CPSC 223, 323, and 365 or 366

**Distribution of courses** B.S. — 6 addtl intermediate or advanced Comp Sci courses; B.A. — four addtl intermediate or advanced Comp Sci courses

**Substitution permitted** Advanced courses in other depts, with DUS permission

**Senior requirement** Senior project (CPSC 490)
FACULTY OF THE DEPARTMENT OF COMPUTER SCIENCE


Associate Professor Mahesh Balakrishnan


Senior Research Scientists Robert Bjornson, Andrew Sherman

Senior Lecturer Stephen Slade

Lecturers Benedict Brown, James Glenn, Kyle Jensen, *Natalie Melo, Scott Petersen, Brad Rosen, Xiyan Tang

*A secondary appointment with primary affiliation in another department or school.
†A joint appointment with primary affiliation in another department or school.

For a complete list of Computer Science Department personnel, visit http://cpsc.yale.edu/.

Introductory Courses

* CPSC 035b, Twenty-First Century Electronic and Computer Music Techniques Scott Petersen
Exploration of twenty-first century electronic and computer music through the diverse subjects and issues at the intersection of technology and new music. How computers have changed and challenged the analysis, composition, production, and appreciation of music over the last fifty years. Knowledge of basic music theory and the ability to read Western musical notation is assumed. Enrollment limited to first-year students. Preregistration required; see under First-Year Seminar Program.

CPSC 100a / CPSC S100, Introduction to Computing and Programming Benedict Brown
Introduction to the intellectual enterprises of computer science and to the art of programming. Students learn how to think algorithmically and solve problems efficiently. Topics include abstraction, algorithms, data structures, encapsulation, resource management, security, software engineering, and web development. Languages include C, Python, SQL, and JavaScript, plus CSS and HTML. Problem sets inspired by real-world domains of biology, cryptography, finance, forensics, and gaming. See CS50's website, https://cs50.yale.edu, for additional information. No previous programming experience required. Open to students of all levels and majors.

CPSC 101b, Great Ideas in Computer Science Stanley Eisenstat
An introduction for nonmajors to some of the most important ideas in computer science: what the computer is; how it works; what it can do and what it cannot do, now and in the future. Topics include algorithms, elementary programming, hardware, language interpretation, software engineering, complexity, models of computation, and artificial intelligence. No previous programming experience required. QR

CPSC 112b, Introduction to Programming Benedict Brown
Development on the computer of programming skills, problem-solving methods, and selected applications. No previous experience with computers necessary. QR

CPSC 134a / MUSI 426a, Programming Musical Applications Scott Petersen
Topics in computer music, including musical representations for computing, automated music analysis and composition, interactive systems, and virtual instrument design. Use of domain-specific programming languages and libraries to explore how the principles of computer science can be applied to music to create new interfaces, instruments, and tools. Recommended preparation: the ability to read music or play an instrument. QR

* CPSC 150a, Computer Science and the Modern Intellectual Agenda David Gelernter
Introduction to the basic ideas of computer science (computability, algorithm, virtual machine, symbol processing system), and of several ongoing relationships between computer science and other fields, particularly philosophy of mind. No previous experience with computers necessary. Enrollment limited to 25. WR, HU

CPSC 183a, Law, Technology, and Culture Brad Rosen
An exploration of the myriad ways in which law and technology intersect, with a special focus on the role of cyberspace. Topics include digital copyright, fair speech, privacy and anonymity, information security, innovation, online communities, the impact of technology on society, and emerging trends. No previous experience with computers or law necessary. SO

* CPSC 184b, Intellectual Property in the Digital Age Xiyan Tang
The evolving and oftentimes vexing intellectual property regime of the new digital age. Focus on copyright, fair use, remix culture, access to knowledge, technological innovations, the increasing relevance of trademarks in the new information society, the tension between
creativity/creating and the intellectual property rules which either foster or inhibit it, and the new information culture of the digital age. Prerequisite: CPSC 183 or permission of instructor. HU, SO

* CPSC 185b, Control, Privacy, and Technology  Brad Rosen
The evolution of various legal doctrines with and around technological development. Topics include criminal law, privacy, search and seizure, digital rights, and the implications of technologically permitted methods of control on the law. Special attention to case law and policy. After CPSC 183. WR, SO

CPSC 200b, Introduction to Information Systems  Stephen Slade
The real-world artifacts and implementations that comprise the vital computational organisms that populate our world. Hardware and software and the related issues of security, privacy, regulation, and software engineering. Examples stress practical applications of technology, as well as limitations and societal issues. After CPSC 100 or 112 or equivalent. QR

CPSC 201a or b, Introduction to Computer Science  Stephen Slade
Introduction to the concepts, techniques, and applications of computer science. Topics include computer systems (the design of computers and their languages); theoretical foundations of computing (computability, complexity, algorithm design); and artificial intelligence (the organization of knowledge and its representation for efficient search). Examples stress the importance of different problem-solving methods. After CPSC 112 or equivalent. QR

CPSC 202a, Mathematical Tools for Computer Science  Dana Angluin
Introduction to formal methods for reasoning and to mathematical techniques basic to computer science. Topics include propositional logic, discrete mathematics, and linear algebra. Emphasis on applications to computer science: recurrences, sorting, graph traversal, Gaussian elimination. QR

CPSC 223a or b, Data Structures and Programming Techniques  James Glenn
Topics include programming in C; data structures (arrays, stacks, queues, lists, trees, heaps, graphs); sorting and searching; storage allocation and management; data abstraction; programming style; testing and debugging; writing efficient programs. After CPSC 201 or equivalent. QR RP

CPSC 257a, Information Security in the Real World  Stephen Slade
Introduction to information security, the practice of protecting information from unauthorized actions, in the context of computer systems. Topics include current security-related issues, basic adversarial models and threats to computer systems, potential defenses, security tools, and common security breaches and their wider impacts. Prerequisite: CPSC 100, 112, or equivalent programming experience, or with permission of instructor. QR

CPSC 276a, Introduction to Web Application for the Digital Humanities  Benedict Brown
Introduction to applications of computer and data science in the humanities, including web technologies, visualization, and database design. Students work in teams to develop a variety of applications proposed by faculty and staff from the Digital Humanities Lab, the Institute for the Preservation of Cultural Heritage, and the Computer Science department. Meets with CPSC 376. Students may earn credit for CPSC 276 or 376; not both. Prerequisite: CPSC 110, CPSC 112, equivalent programming experience, or permission of the instructor. QR

* CPSC 290a or b, Directed Research  Staff
Individual research. Requires a faculty supervisor and the permission of the director of undergraduate studies. May be taken more than once for credit.

Intermediate Courses

CPSC 323a, Introduction to Systems Programming and Computer Organization  Stanley Eisenstat
Machine architecture and computer organization, systems programming in a high-level language, issues in operating systems, software engineering, prototyping in scripting languages. After CPSC 223. QR RP

CPSC 338b / EENG 348b, Digital Systems  Rajit Manohar
Development of engineering skills through the design and analysis of digital logic components and circuits. Introduction to gate-level circuit design, beginning with single gates and building up to complex systems. Hands-on experience with circuit design using computer-aided design tools and programmable logic devices. Recommended preparation: EENG 201b. RP

* CPSC 365b, Intensive Algorithms  Staff
Mathematically sophisticated treatment of the design and analysis of algorithms and the theory of NP completeness. Algorithmic paradigms including greedy algorithms, divide and conquer, dynamic programming, network flow, approximation algorithms, and randomized algorithms. Problems drawn from the social sciences, Data Science, Computer Science, and engineering. For students with a flair for proofs and problem solving. Either CPSC 365 or CPSC 366 may be taken for credit. Prerequisites: MATH 244 and CPSC 223. QR

CPSC 366b, Algorithms  James Glenn
Paradigms for algorithmic problem solving: greedy algorithms, divide and conquer, dynamic programming, and network flow. NP completeness and approximation algorithms for NP-complete problems. Algorithms for problems from economics, scheduling, network design and navigation, geometry, biology, and optimization. Provides algorithmic background essential to further study of computer science. Either CPSC 365 or CPSC 366 may be taken for credit. Prerequisites: CPSC 202 and 223. QR
* CPSC 376a, Advanced Web Application Development in the Digital Humanities  Benedict Brown
Advanced applications of computer and data science in the humanities, including web technologies, visualization, and database design. Students work in teams to develop a variety of applications proposed by faculty and staff from the Digital Humanities Lab, the Institute for the Preservation of Cultural Heritage, and the Computer Science department. Meets with CPSC 376. Students may earn credit for CPSC 276 or 376; not both. Prerequisite: CPSC 223 or equivalent, or permission of the instructor.  QR

Advanced Courses

**CPSC 422a, Design and Implementation of Operating Systems**  Zhong Shao
The design and implementation of operating systems. Topics include synchronization, deadlock, process management, storage management, file systems, security, protection, and networking. After CPSC 323.  QR

**CPSC 423b, Principles of Operating Systems**  Abraham Silberschatz
A survey of the underlying principles of modern operating systems. Topics include process management, memory management, storage management, protection and security, distributed systems, and virtual machines. Emphasis on fundamental concepts rather than implementation. After CPSC 323.  QR

**CPSC 424a, Parallel Programming Techniques**  Andrew Sherman
Practical introduction to parallel programming, emphasizing techniques and algorithms suitable for scientific and engineering computations. Aspects of processor and machine architecture. Techniques such as multithreading, message passing, and data parallel computing using graphics processing units. Performance measurement, tuning, and debugging of parallel programs. Parallel file systems and I/O. After CPSC 223 and MATH 222 or 225, or equivalents.  QR

[ CPSC 426, Building Distributed Systems ]

**CPSC 427a, Object-Oriented Programming**  Michael Fischer
Object-oriented programming as a means to efficient, reliable, modular, reusable code. Use of classes, derivation, templates, name-hiding, exceptions, polymorphic functions, and other features of C++. After CPSC 223.  QR

[ CPSC 428, Language-Based Security ]

[ CPSC 430, Formal Semantics ]

**CPSC 431a / MUSI 428a, Computer Music: Algorithmic and Heuristic Composition**  Scott Petersen
Study of the theoretical and practical fundamentals of computer-generated music, with a focus on high-level representations of music, algorithmic and heuristic composition, and programming languages for computer music generation. Theoretical concepts are supplemented with pragmatic issues expressed in a high-level programming language. Ability to read music is assumed. After CPSC 202 and 223.  QR

**CPSC 432b, Computer Music: Sound Representation and Synthesis**  Scott Petersen
Study of the theoretical and practical fundamentals of computer-generated music, with a focus on low-level sound representation, acoustics and sound synthesis, scales and tuning systems, and programming languages for computer music generation. Theoretical concepts are supplemented with pragmatic issues expressed in a high-level programming language. Ability to read music is assumed. After CPSC 202 and 223.  QR

**CPSC 433a, Computer Networks**  Yang Yang
An introduction to the design, implementation, analysis, and evaluation of computer networks and their protocols. Topics include layered network architectures, applications, transport, congestion, routing, data link protocols, local area networks, performance analysis, multimedia networking, network security, and network management. Emphasis on protocols used in the Internet. After CPSC 323.  QR

[ CPSC 434b, Topics in Networked Systems ]

[ CPSC 436, Networked Embedded Systems and Sensor Networks ]

**CPSC 437a, Introduction to Database Systems**  Abraham Silberschatz

[ CPSC 438, Database System Implementation and Architectures ]

**CPSC 439b, Software Engineering**  Ruzica Piskac
Introduction to fundamental concepts in software engineering and to the development and maintenance of large, robust software systems. The process of collecting requirements and writing specifications; project planning and system design; methods for increasing
software reliability, including delta debugging and automatic test-case generation; type systems, static analysis, and model checking. Students build software in teams. After CPSC 323. QR RP

* CPSC 451b, The User Interface  David Gelernter
The user interface (UI) in the context of modern design, where tech has been a strong and consistent influence from the Bauhaus and U.S. industrial design of the 1920s and 1930s through the IBM-Eames design project of the 1950s to 1970s. The UI in the context of the windows-menu-mouse desktop, as developed by Alan Kay and Xerox in the 1970s and refined by Apple in the early 1980s. Students develop a detailed design and simple implementation for a UI. Prerequisite: CPSC 223 or equivalent.

CPSC 453a, Machine Learning for Biology  Smita Krishnaswamy
Applications of machine learning methods in the analysis of high-throughput biological data with focus on genomic and proteomic data. Topics include methods for denoising data; non-linear dimensionality reduction for visualization and progression analysis; unsupervised clustering; and information theoretic analysis of gene regulatory and signaling networks.

CPSC 454a, Software Analysis and Verification  Ruzica Piskac
Introduction to concepts, tools, and techniques used in the formal verification of software. State-of-the-art tools used for program verification; detailed insights into algorithms and paradigms on which those tools are based, including model checking, abstract interpretation, decision procedures, and SMT solvers. After CPSC 202 and 323 or equivalents. QR RP

[ CPSC 455, Economics and Computation ]

* CPSC 456b / EENG 451b, Wireless Technologies and the Internet of Things  Wenjun Hu
Fundamental theory of wireless communications and its application explored against the backdrop of everyday wireless technologies such as WiFi and cellular networks. Channel fading, MIMO communication, space-time coding, opportunistic communication, OFDM and CDMA, and the evolution and improvement of technologies over time. Emphasis on the interplay between concepts and their implementation in real systems. Prerequisites: 1) Introductory courses in mathematics, engineering, or computer science covering basics of the following topics: Linux skills, Matlab programming, probability, linear algebra, and Fourier transform; 2) Or by permission of the instructor. The course material will be self-contained as much as possible. The labs and homework assignments require Linux and Matlab skills and simple statistical and matrix analysis (using built-in Matlab functions). There will be a couple of introductory labs to refresh Linux and Matlab skills if needed.

[ CPSC 457, Sensitive Information in a Connected World ]
[ CPSC 462, Graphs and Networks ]

CPSC 465b, Theory of Distributed Systems  James Aspnes
Models of asynchronous distributed computing systems. Fundamental concepts of concurrency and synchronization, communication, reliability, topological and geometric constraints, time and space complexity, and distributed algorithms. After CPSC 365 or 366. QR

CPSC 467a / CPSC 367b, Cryptography and Computer Security  Mariana Raykova
A survey of such private and public key cryptographic techniques as DES, RSA, and zero-knowledge proofs, and their application to problems of maintaining privacy and security in computer networks. Focus on technology, with consideration of such societal issues as balancing individual privacy concerns against the needs of law enforcement, vulnerability of societal institutions to electronic attack, export regulations and international competitiveness, and development of secure information systems. Some programming may be required. After CPSC 202 and 223. QR

CPSC 468a, Computational Complexity  Joan Feigenbaum
Introduction to the theory of computational complexity. Basic complexity classes, including polynomial time, nondeterministic polynomial time, probabilistic polynomial time, polynomial space, logarithmic space, and nondeterministic logarithmic space. The roles of reductions, completeness, randomness, and interaction in the formal study of computation. After CPSC 365 or 366, or with permission of instructor. QR

[ CPSC 469, Randomized Algorithms ]

CPSC 470b, Artificial Intelligence  Brian Scassellati
Introduction to artificial intelligence research, focusing on reasoning and perception. Topics include knowledge representation, predicate calculus, temporal reasoning, vision, robotics, planning, and learning. After CPSC 201 and 202. QR

[ CPSC 471, Advanced Topics in Artificial Intelligence ]

* CPSC 472a, Intelligent Robotics Laboratory  Brian Scassellati
Students work in small teams to construct novel research projects using one of a variety of robot architectures. Project topics may include human-robot interaction, adaptive intelligent behavior, active perception, humanoid robotics, and socially assistive robotics. Enrollment limited to 20. After CPSC 472. QR

CPSC 474a, Computational Intelligence for Games  James Glenn
Introduction to techniques used for creating computer players for games, particularly board games. Topics include combinatorial and classical game theory, stochastic search methods, applications of neural networks, and procedural content generation. Prerequisites: CPSC 202 and CPSC 223. QR
An overview of computational vision with a biological emphasis. Suitable as an introduction to biological perception for computer science and engineering students, as well as an introduction to computational vision for mathematics, psychology, and physiology students.

Prerequisite: CPSC 112 and MATH 120, or with permission of instructor. QP, SC RP

Advanced view of vision from a mathematical, computational, and neurophysiological perspective. Emphasis on differential geometry, machine learning, visual psychophysics, and advanced neurophysiology. Topics include perceptual organization, shading, color and texture analysis, and shape description and representation. After CPSC 475. QP, SC

Linguistic, mathematical, and computational fundamentals of natural language processing (NLP). Topics include part of speech tagging, Hidden Markov models, syntax and parsing, lexical semantics, compositional semantics, machine translation, text classification, discourse, and dialogue processing. Additional topics such as sentiment analysis, text generation, and deep learning for NLP.

Prerequisites: CPSC 202 and CPSC 223, or permission of instructor. QP

Introduction to the basic concepts of two- and three-dimensional computer graphics. Topics include affine and projective transformations, clipping and windowing, visual perception, scene modeling and animation, algorithms for visible surface determination, reflection models, illumination algorithms, and color theory. After CPSC 202 and 223. QP

An in-depth study of advanced algorithms and systems for rendering, modeling, and animation in computer graphics. Topics vary and may include reflectance modeling, global illumination, subdivision surfaces, NURBS, physically-based fluids systems, and character animation. After CPSC 202 and 223. QP

Individual study for qualified students who wish to investigate an area of computer science not covered in regular courses. A student must be sponsored by a faculty member who sets the requirements and meets regularly with the student. Requires a written plan of study approved by the faculty adviser and the director of undergraduate studies. May be taken more than once for credit.

Individual research intended to fulfill the senior requirement. Requires a faculty supervisor and the permission of the director of undergraduate studies. The student must submit a written report about the results of the project.