Computer Science

Computing has grown to be a pervasive element of society, business, science, and entertainment. The availability of relatively inexpensive high performance computing capabilities, ubiquitous high speed and wireless networking, and mobile computing have powered a technology-driven restructuring of the way society and most professions now operate. Information, and its associated processing and transport, is the commodity upon which corporations are built and lives are improved. At the center of this revolution, making it happen, are those who study computer science.

There are two dimensions to the field of computer science that establish it as a unique area. CS can be viewed as a stand-alone discipline worthy of study unto itself, and/or as an empowering discipline to be studied in conjunction with other areas. Core CS careers include (but are not limited to) software design and development, computer systems engineering or administration, and information security. Application areas span a wide range of fields and disciplines such as robotics, medical or health informatics, gaming/entertainment, business computing, and scientific research to name a few.

Because computer science is a highly diverse and broadly applied field, studies can proceed in many different directions. Accordingly, the undergraduate and graduate programs in the Department of Computer Science at Johns Hopkins are flexible curricula designed to accommodate a wide range of goals. A student at Johns Hopkins can pursue appropriately customized versions of the following computer science programs: minor, bachelor of science, bachelor of arts, masters of science in engineering, and doctor of philosophy. Most of this catalog section is devoted to details regarding these programs.

Computer science research laboratories are currently active in the following areas at Johns Hopkins: algorithm design and analysis, human-computer interaction, machine learning, health informatics, computational medicine, computer vision and image processing, computer graphics, geometric modeling, programming languages, natural language and speech processing, information retrieval, cryptography and information security, secure and robust systems, storage systems, high-performance and scientific computing, computational genomics, networks and distributed systems, stream processing, parallel and distributed databases, robotics, computer-integrated surgical systems, and wireless and sensor systems.

Additionally, interdisciplinary research centers in the university have heavy involvement by Computer Science faculty: the Information Security Institute (ISI), the Center for Computer-Integrated Surgical Systems and Technology (CISST), the Laboratory for Computational Sensing and Robotics (LCSR), the Center for Language and Speech Processing (CLSP), and the Institute for Data Intensive Engineering and Science (IDIES). An important component of the educational process in the department is the opportunity for undergraduate and graduate student participation in the research programs of the faculty. In particular, original research in close association with individual faculty members is emphasized at the graduate level.

There are several closely related programs at the undergraduate and graduate levels which involve significant coursework and faculty involvement from the Department of Computer Science. The Laboratory for Computational Sensing and Robotics (LCSR) offers a minor in robotics and also a minor in computer integrated surgery through the Engineering Research Center for Computer Integrated Surgical Systems and Technology. Details of these programs may be found elsewhere in this catalog in the section pertaining to the Laboratory for Computational Sensing and Robotics (http://e-catalog.jhu.edu/departments-program-requirements-and-courses/engineering/robotics-computational-sensing). Undergraduates with a strong interest in system design and performance may elect to pursue a bachelor degree in computer engineering (http://e-catalog.jhu.edu/departments-program-requirements-and-courses/engineering/electrical-computer-engineering). This field of study includes course work in computer science, as well as electrical and computer engineering. Although jointly administered by both departments, specific goals and requirements of the computer engineering degree may be found in the catalog section pertaining to the Department of Electrical and Computer Engineering only.

At the graduate level, the LCSR (http://e-catalog.jhu.edu/departments-program-requirements-and-courses/engineering/robotics-computational-sensing) offers a Master of Science in Engineering (M.S.E.) in Robotics, designed for students from a wide variety of engineering, scientific, and mathematical backgrounds to advance their interdisciplinary knowledge in robotics. Details of this program may be found in the LCSR section of this catalog, or on the web at www.lcsr.jhu.edu/MSE. Lastly, the Master of Science in Security Informatics (M.S.S.I.) is a specialized graduate program offered through the Information Security Institute (http://e-catalog.jhu.edu/departments-program-requirements-and-courses/engineering/information-security-institute) (ISI) in the WSE. The field of security informatics is fundamentally based on information security and assurance technologies (hardware, software, and networks) as related to issues such as policy, management, privacy/trust, health care, and law, from both national and international perspectives. Interested students can obtain detailed information regarding the M.S.S.I. online at www.jhuisi.jhu.edu or in the ISI section of this catalog.

For additional information regarding the academic programs available in Computer Science, and the facilities provided, please consult the sections which follow, or the departmental website www.cs.jhu.edu.

Facilities

The CS department is primarily housed in Malone Hall, a state of the art, open concept research facility. Additional department research space is located in the adjacent Hackerman Hall.

The general department computing facilities include numerous workstations and servers. Undergraduate laboratories combine to provide approximately 24 Linux workstations, and several Windows stations. One of these labs is a collaboration room allowing students to work in a team-based environment, with several private breakout rooms as well. At the graduate level there is a Master’s Lab consisting of a collaboration area and workstation area, both consisting of several Linux workstations. All Ph.D. students are assigned dedicated desks in their research labs. Additionally, multiple high-speed networked laser printers, as well as a networked color copier and remotely accessible Linux compute servers are available to both graduate and undergraduate students.

Focused research laboratories have significant resources that provide greater specialization, including isolated networks of PCs for security studies, high-performance computing clusters, sensor and wireless computing testbeds, robots and computer vision systems, a mock operating room equipped with medical robots and imaging equipment, and more.
The general department computing facilities are tied together by our own LAN, and access to specialized hardware in other departments, labs, and institutions is available via the university intranet and the Internet. In addition, the university provides wireless access to the JHU intranet and the Internet, as well as server systems that provide e-mail accounts for all students.

(See also General Requirements for Departmental Majors (http://e-catalog.jhu.edu/undergrad-students/academic-policies/requirements-for-a-bachelors-degree))

The objectives of our bachelor degree programs are to train computer scientists who will be able to:

• Successfully engage in professional practice in the computing sciences or apply computer science tools and techniques to another field of interest.
• Pursue advanced study in the computing sciences.
• Behave in a professional and ethical manner.
• Work successfully in both independent and team environments.

A successful major program of study leads to either the Bachelor of Science in Computer Science (B.S.) or the Bachelor of Arts in Computer Science (B.A.). Students should decide which degree program to complete by about their junior year. Both degree programs require specific courses and/or credits in several key areas: computer science, math, basic science, humanities and social sciences. However, there is much flexibility in how these requirements are fulfilled. Undergraduate majors may choose to pursue a broad selection of computer science and distributional courses, or to pursue a specific track within the field. Current tracks reflect departmental and school strengths: information security, natural language processing, robotics, software engineering, and video game design. Further information on these tracks may be found in the computer science undergraduate advising manual.

All undergraduate students majoring or minoring in computer science must have a faculty advisor in the department. They will be assigned an advisor as entering freshmen or upon deciding on the major/minor. Every major must follow a program approved by his/her faculty advisor.

The department also offers a minor in computer science, and tangentially, a minor in computer integrated surgery and a minor in robotics. Some students majoring in computer science may be eligible for a concurrent bachelor’s/master’s degree program. Requirements for these programs are included here as well. Additional details regarding undergraduate programs can be found in the department’s undergraduate advising manual or on the website at www.cs.jhu.edu.

Requirements for the B.S. Degree

The Bachelor of Science in Computer Science degree program is accredited by the Computing Accreditation Commission of ABET, www.abet.org (http://www.abet.org). It provides for the acquisition of the following knowledge base and skill set:

• An ability to apply knowledge of computing and mathematics appropriate to the discipline.
• An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
• An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
• An ability to function effectively on teams to accomplish a common goal.
• An understanding of professional, ethical, legal, security, and social issues and responsibilities.
• An ability to communicate effectively with a range of audiences.
• An ability to analyze the local and global impact of computing on individuals, organizations, and society.
• Recognition of the need for and an ability to engage in continuing professional development.
• An ability to use current techniques, skills, and tools necessary for computing practice.
• An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the trade-offs involved in design choices.
• An ability to apply design and development principles in the construction of software systems of varying complexity.

To meet the course credit requirements for the B.S. in computer science, the student must complete a minimum of 126 credits. The basic requirements for the B.S. degree are as follows:

<table>
<thead>
<tr>
<th>Course</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science</td>
<td>42</td>
</tr>
<tr>
<td>Mathematics</td>
<td>22</td>
</tr>
<tr>
<td>Basic Sciences</td>
<td>16</td>
</tr>
<tr>
<td>Humanities/Social Sciences</td>
<td>18</td>
</tr>
<tr>
<td>Broadening Electives</td>
<td>12</td>
</tr>
<tr>
<td>Two Writing Intensive Courses</td>
<td>16</td>
</tr>
<tr>
<td>Electives</td>
<td></td>
</tr>
<tr>
<td>Total Credits</td>
<td>126</td>
</tr>
</tbody>
</table>

Details and course recommendations for these distributional requirements are below. These requirements add up to at most 116 credits and fulfill general university requirements, leaving at least 10 elective credits that students choose freely. Freshman majors are also expected to take EN.600.105 M & Ms: Freshman Experience (optional for transfers into the major). This is a one credit S/U course that may only be counted as an elective.

Except for electives, courses should not be taken on a satisfactory/unsatisfactory basis. By university policy, no more than 18 D or D+ credits can be counted toward the total credit requirements for a degree.

Computer Science (42 credits)

The following foundational courses in computer science must be included in a student’s program:

EN.600.104 Computer Ethics 1
EN.600.107 Introductory Programming in Java (or equivalent) 3
EN.600.120 Intermediate Programming 4
EN.600.226 Data Structures 4
EN.600.233 Computer System Fundamentals 3
EN.600.271 Automata & Computation Theory 3
EN.600.363 Introduction To Algorithms 3
or EN.600.463 Algorithms I
At least 16 credit hours must be at the 300-level or above, including EN.600.363/463. At least one course in each classification area of Analysis, Applications and Systems must be chosen. An exhaustive list of the area classifications for each of our courses may be found on the department’s website.

Students must take at least one of the following courses which contain oral communication components. The course satisfying this requirement may overlap other requirements:

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN.600.255</td>
<td>Introduction to Video Game Design</td>
</tr>
<tr>
<td>EN.600.321/421</td>
<td>Object Oriented Software Engineering</td>
</tr>
<tr>
<td>EN.600.355</td>
<td>Video Game Design Project</td>
</tr>
<tr>
<td>EN.600.392</td>
<td>CS Design Project</td>
</tr>
<tr>
<td>EN.600.446</td>
<td>Computer Integrated Surgery II</td>
</tr>
<tr>
<td>EN.600.520</td>
<td>Senior Honors Thesis</td>
</tr>
</tbody>
</table>

Eight additional credits of Computer Science are required.

**Mathematics (22 credits)**
The following courses must be included:

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS.110.108</td>
<td>Calculus I</td>
</tr>
<tr>
<td>AS.110.109</td>
<td>Calculus II (For Physical Sciences and Engineering)</td>
</tr>
<tr>
<td>EN.550.171</td>
<td>Discrete Mathematics</td>
</tr>
</tbody>
</table>

The remaining courses must be 200-level or above, chosen from Mathematics (110.xxx) or Applied Math and Statistics (550.xxx), and must include coverage of both probability and statistics. Some highly recommended math electives are Probability & Statistics, Calculus III, Linear Algebra, and Differential Equations. Note that students will need at least six courses to fulfill the credit requirement. **

**Basic Sciences (16 credits)**
At least two semesters of physics or two semesters of chemistry, with the associated laboratories, must be included. The remaining courses must be chosen in accordance with the list posted on the department’s website, which includes most courses in Physics, Chemistry, Biology, Biophysics, Earth & Planetary Sciences, Natural Sciences designated engineering courses, and some Natural Sciences designated courses in Neuroscience & Cognitive Science, but not all. At most 2 credits from satisfactory/unsatisfactory intersession courses may be used to fulfill this requirement.

**Humanities/Social Sciences (18 credits)**
Six courses in the Humanities and Social and Behavioral Sciences must be taken, with each course at least 3 credits. These courses must have either an Humanities or Social and Behavioral Sciences area designator on them; however, foreign language courses (without an Humanities or Social and Behavioral Sciences) may also be used to satisfy this requirement.

**Broadening Electives (12 credits)**
At least 12 additional credits must be taken in areas such as Humanities, Social and Behavioral Science, Natural Science, arts, business, engineering, or other disciplines that serve to broaden the student’s background. Courses used to fulfill this requirement must not be from computer science, computer and electrical engineering, or math departments.

**Writing Requirement**
Students are required to fulfill the university’s requirement of two writing intensive courses, each at least 3 credits. Students must receive at least a C- grade or better in these writing courses. These courses may overlap other requirements. At least one of the following options should be chosen:

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS.060.113</td>
<td>Expository Writing</td>
</tr>
<tr>
<td>EN.661.110</td>
<td>Professional Writing and Communication</td>
</tr>
<tr>
<td>AS.220.105</td>
<td>Fiction Poetry Writing I</td>
</tr>
<tr>
<td>or AS.220.106</td>
<td>Fiction Poetry Writing II</td>
</tr>
</tbody>
</table>

**Electives**
Electives to be chosen by the student with guidance and approval of his/her advisor.

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN.600.105</td>
<td>M &amp; Ms: Freshman Experience</td>
</tr>
<tr>
<td>EN.600.110</td>
<td>Fiction Poetry Writing I</td>
</tr>
<tr>
<td>EN.600.171</td>
<td>Fiction Poetry Writing I</td>
</tr>
<tr>
<td>EN.661.110</td>
<td>Professional Writing and Communication</td>
</tr>
</tbody>
</table>

**Total Credits** 126

* With the advisor’s explicit permission regarding course selections, up to 6 of the 42 required credits may be taken in the Department of Electrical and Computer Engineering or the Information Security Institute.

No more than 6 credits of independent study (including EN.600.491-EN.600.492) and no more than 3 credits of short courses can be counted toward this requirement. However, B.S. students doing the Senior Honors Thesis (EN.600.519-EN.600.520) may use an additional three credits of independent work toward their CS requirements, for a total of 9 credits.

No courses with grades below C- or with satisfactory/unsatisfactory grades can be used to fulfill this requirement unless they are not offered for a grade.

** Requirements for the B.A. Degree**
To meet the course credit requirements for the B.A. in computer science, the student must complete a minimum of 120 credits. The basic requirements for the B.A. degree are:

<table>
<thead>
<tr>
<th>Department</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science</td>
<td>32</td>
</tr>
<tr>
<td>Mathematics</td>
<td>18</td>
</tr>
<tr>
<td>Basic Sciences</td>
<td>12</td>
</tr>
<tr>
<td>Humanities/Social Sciences</td>
<td>18</td>
</tr>
<tr>
<td>Foreign Language</td>
<td>6</td>
</tr>
<tr>
<td>4 Writing Intensive Courses</td>
<td></td>
</tr>
<tr>
<td>Electives</td>
<td>34</td>
</tr>
</tbody>
</table>

**Total Credits** 120

* To be chosen by the student with the guidance and approval of his/her advisor. Freshman majors are also expected to take EN.600.105 M & Ms: Freshman Experience (optional for transfers into the major). This is a one credit Satisfactory/Unsatisfactory course that may only be counted as an elective.

Details and course recommendations of these distributional requirements are below. These requirements add up to 84 credits and fulfill general university distribution requirements.

Except for electives, courses should not be taken on a satisfactory/unsatisfactory basis. By university policy, no more than 18 D or D
+ credits can be counted toward the total credit requirements for a degree.

**Computer Science (32 credits)**

The following foundational courses in computer science must be included in a student’s program:

- EN.600.107 Introductory Programming in Java (or equivalent) 3 credits
- EN.600.120 Intermediate Programming 4 credits
- EN.600.226 Data Structures 4 credits
- EN.600.233 Computer System Fundamentals 3 credits
- EN.600.271 Automata & Computation Theory 3 credits
- EN.600.363 Introduction To Algorithms 3 credits or EN.600.463 Algorithms I

At least 15 credit hours must be at the 300-level or above, including 12 EN.600.363/463.

**Mathematics (18 credits)**

The following courses must be included:

- AS.110.108 Calculus I 4 credits
- AS.110.109 Calculus II (For Physical Sciences and Engineering) 4 credits
- EN.550.171 Discrete Mathematics 4 credits

The remaining courses may be chosen from Mathematics (110.xxx) or Applied Math and Statistics (550.xxx). At least one course must be 200-level or above. **

**Basic Sciences (12 credits)**

At least two semesters of physics or chemistry or a combination of both, with the associated laboratories, must be included. The remaining courses must be chosen in accordance with the list posted on the department website, which includes most courses in Physics, Chemistry, Biology, Biophysics, Earth & Planetary Sciences, ‘N’ designated engineering courses, and some ‘N’ designated courses in Neuroscience and Cognitive Science, but not all. At most 2 credits from (S/U) intersession courses may be used to fulfill this requirement.

**Humanities/Social Sciences (18 credits)**

Six courses in the Humanities/Social Sciences must be taken, with each course at least 3 credits. At least two 3-credit courses at the 300-level or above are required. As befits a B.A. degree, students have ample flexibility to choose courses that broaden the scope of their study, in consultation with their advisors. A subset of the courses selected to satisfy this requirement should demonstrate coherence within an area. Any course with Humanities or Social Sciences area designators may fulfill these distributional requirements.

**Foreign Language (6 credits)**

Two courses in a foreign language, with a total of at least 6 credits are required. These foreign language credits are in addition to the 18 required Humanities/Social Sciences credits.

**Writing Requirement**

All primary computer science majors pursuing a B.A. degree are required to fulfill the university’s requirement of four writing intensive courses, each at least 3 credits. Students must receive at least a C- grade in these courses. These courses may overlap other requirement areas. Highly recommended, at least one of:

- AS.060.113 Expository Writing
- AS.220.105 Fiction Poetry Writing I & AS.220.106 and Fiction Poetry Writing II

---

**Minor in Computer Science**

To satisfy the course credit requirements for a minor in computer science, a student must take a minimum of seven courses, with a total of at least 23 credits, earning at least a C- in each course. These must include four core courses, to provide the student with a foundation, and three upper-level courses (300-level and above), to allow the student to pursue an advanced topic in depth.

While not explicitly required, we also strongly recommend taking EN.550.171 Discrete Mathematics as preparation for several computer science courses, including EN.600.271 Automata & Computation Theory.

**Core Courses**

<table>
<thead>
<tr>
<th>Course</th>
<th>Title</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN.600.107</td>
<td>Introductory Programming in Java (or AP credit)</td>
<td>3</td>
</tr>
<tr>
<td>EN.600.120</td>
<td>Intermediate Programming</td>
<td>4</td>
</tr>
<tr>
<td>EN.600.226</td>
<td>Data Structures</td>
<td>4</td>
</tr>
<tr>
<td>EN.600.233</td>
<td>Computer System Fundamentals</td>
<td>3</td>
</tr>
<tr>
<td>EN.600.271</td>
<td>Automata &amp; Computation Theory</td>
<td>3</td>
</tr>
</tbody>
</table>

Choose one of the following:

- EN.600.233 Computer System Fundamentals 4 credits
- EN.600.271 Automata & Computation Theory 3 credits

**Upper-Level Courses**

These courses should be chosen to form a cohesive minor and must be accepted by the computer science minor advisor. **

<table>
<thead>
<tr>
<th>Course</th>
<th>Title</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN.600.107</td>
<td>Introductory Programming in Java (or AP credit)</td>
<td>3</td>
</tr>
<tr>
<td>EN.600.120</td>
<td>Intermediate Programming</td>
<td>4</td>
</tr>
<tr>
<td>EN.600.226</td>
<td>Data Structures</td>
<td>4</td>
</tr>
<tr>
<td>EN.600.233</td>
<td>Computer System Fundamentals</td>
<td>3</td>
</tr>
<tr>
<td>EN.600.271</td>
<td>Automata &amp; Computation Theory</td>
<td>3</td>
</tr>
</tbody>
</table>

Total Credits 23

**Electives**

Electives to be chosen by the student with guidance and approval of his/her advisor.

<table>
<thead>
<tr>
<th>Course</th>
<th>Title</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN.661.110</td>
<td>Professional Writing and Communication</td>
<td>120</td>
</tr>
</tbody>
</table>

**Total Credits** 120

* With the advisor’s permission, up to 3 of the 32 required credits may be taken in the Department of Electrical and Computer Engineering or the Information Security Institute.

No more than 3 credits of short courses or 3 credits of independent study (including EN.600.491-EN.600.492) may be applied toward this requirement. However, B.A. students doing the Senior Honors Thesis (EN.600.519-EN.600.520) may use an additional 3 credits of independent work toward their CS requirements, for a total of 6 credits.

No courses with grades below C- or with satisfactory/unsatisfactory grades may be used to fulfill this requirement unless they are not offered for a grade.

** Highly recommended: Calculus III, Linear Algebra, Differential Equations, Probability/Statistics. Note that students will need at least five courses to fulfill the credit requirement.

** It is strongly recommended that students choose all three substitutions for these core courses are possible.

Short courses cannot be used toward the minor requirements. All courses must be taken for a grade, not Satisfactory/Unsatisfactory.
Students whose primary major is in the Whiting School may use the same courses to satisfy the requirements of the primary major and also those of a computer science minor. Students who plan to fulfill requirements for a minor must go to the Department of Computer Science director of undergraduate studies to declare the minor and be advised on course selections, and inform the Office of Academic Advising by the end of their junior year.

Short Courses
The Department of Computer Science offers 1-credit short courses covering a variety of topics in computer science and engineering. The purpose of the short courses is to expose students to topics of current interest in the field of computer science and engineering. Short courses are taught not only by faculty and graduate students in the Department of Computer Science and visiting faculty from other universities, but by individuals from local government or industry who have demonstrable expertise in a given area and are practicing the application of computer science theory and concepts.

Students should be aware that short course offerings are likely to change from year to year, depending on instructor commitments: there is no guarantee that the same course will be available at a later time. Students interested in getting details about a particular short course can contact the instructor through the departmental office. No more than 3 credits of short courses may be applied toward the computer science course credit requirement for the B.S. or the B.A. degrees.

Double Majors with Computer Science
It is possible for students to pursue a double major program in which one of the majors is computer science. The computer science requirements are flexible enough to allow for combination with most majors in the Whiting School of Engineering and the Krieger School of Arts and Sciences. Whether computer science is your primary or secondary major, you will be assigned a faculty advisor in the department. In order to declare a first or second major in computer science, students should see the Director of Undergraduate Studies before the start of senior year. Those students must also inform the Office of Academic Affairs of the Whiting School of Engineering and the Registrar of their double major status. Subject to restrictions set by the department offering a second major, students whose primary major is in the Whiting School may use courses to satisfy both the requirements of the student’s primary major and those of a double major.

Combined Bachelor’s/Master’s Program
As early as the end of their sophomore year, qualified students may apply for admission to a combined bachelor’s/master’s program which combines a B.S. or B.A. degree (in any department) with a master of science in engineering degree in Computer Science. This program allows students to simultaneously pursue both an undergraduate and a graduate degree program of study. Generally, the combined B.S./M.S.E. or B.A./M.S.E. program is accomplished in five years, although some students take more or less time. Applicants are judged on the basis of their performance in courses and their letters of recommendation. Double counting of at most two courses is subject to current WSE requirements (unless more than two course requirements have been satisfied using courses transferred from other institutions). Please refer to the Ph.D. program information for details.

Every graduate student in the Department of Computer Science must follow a program approved by a faculty advisor in the department. The advisor assigned to a student may change, subject to the acceptance of the new advisor.

Requirements for the M.S.E. Degree
The Master of Science in Engineering (M.S.E.) is a day program offered by the Department of Computer Science. Most students complete the program in three full-time semesters. Two consecutive semesters of residence as a full-time graduate student are required. Those interested in part-time evening study should refer to Engineering for Professionals at ep.jhu.edu.

Entering students are expected to have completed a program of study equivalent to that required by the B.S. in computer science. Applicants from other disciplines are expected to have coursework (or equivalent experience) in intermediate programming (C++ and Java), data structures, computer system fundamentals, and automata theory. Upon admission to the Master of Science in Engineering program, a student is assigned a graduate advisor from the Department of Computer Science who must approve the courses to be applied to the M.S.E. degree.

The Department of Computer Science classifies its courses into three sub-areas: Analysis, Applications, and Systems. All M.S.E. candidates must complete at least two graduate courses (3 class hours each) from each of these three areas. Each upper-level course description in this catalog includes its area for reference. A course in multiple areas may only be counted toward one requirement. A current listing of courses with area designators is provided on the departmental website. While this listing includes a few highly relevant courses outside the Department of Computer Science, only one such course may be applied toward the area requirements. M.S.E. students must also complete an additional two elective graduate courses (chosen from any CS area or from closely related departments such as Electrical and Computer Engineering, Cognitive Science, Mathematics, or Applied Mathematics and Statistics) for a total of eight graduate courses. In the Department of Computer Science, graduate courses are 600-level and above, or 400-level courses that are not offered at the 600-level. The coursework program must be approved by the student’s faculty advisor.

In addition to the eight courses, a student must elect one of the following options in order to fulfill the degree requirements:

- Two additional graduate courses in Computer Science, approved as above.
- A supervised research project including an approved project report that will be made publicly available.
- An original, faculty-approved master’s essay, submitted to the Milton S. Eisenhower Library.

By satisfying the Ph.D. qualifying course requirements and the first qualifying project, a student will also satisfy the M.S.E. degree requirements (unless more than two course requirements have been satisfied using courses transferred from other institutions). Please refer to the Ph.D. program information for details.

All M.S.E. degree candidates are encouraged to regularly attend the department seminars. You may enroll in EN.600.601 Computer Science...
Seminar-EN.600.602 Computer Science Seminar; however, these courses may not be counted toward the degree course requirements.

**Course Requirement Details**

- All courses counted toward the M.S.E. degree requirement must be taken at the graduate level. In the Department of Computer Science, graduate courses are 600-level and above, or 400-level courses that are not offered at the 600-level.

- At most, two courses with grades less than B- may be counted toward the coursework requirements. No courses with grades less than C- may be counted.

- The overall grade point average of the courses counted toward the coursework requirements must be a 3.0 or higher (B average).

- At most, two independent study courses can be counted toward the course requirements.

- Other than independent study courses, no courses with grades of P can be counted toward the coursework requirement. Courses with grades of P will not be included in the grade point average calculation.

- One of the courses required for the M.S.E. degree, but only one, can be replaced by 3 credits from comparable short courses.

- A majority of the courses counted toward the degree must be taught in the Department of Computer Science.

- At most, two courses can be transferred from graduate programs of other institutions to be counted toward the degree requirements. Such transfer courses must be approved by the student’s faculty advisor and the department. It is the obligation of the student to provide all necessary data to the Department of Computer Science regarding the course(s) for which transfer credit is being requested.

- A grade of D or F results in probation; a second D or F is cause for being dropped from the program.

- Any master’s student engaged in research for payment or to help meet degree requirements is required to complete Responsible Conduct of Research training. Students receiving payment from NIH training grants or fellowships must take the in person course —AS.360.625 Responsible Conduct of Research. All other students can take the course online—AS.360.624 Responsible Conduct of Research (Online). Instructions for accessing and signing up for the course can be found here: http://engineering.jhu.edu/wse-research/resources-policies-forms/responsible-conduct-of-research/online-training-course-for-the-responsible-conduct-of-research/

- Additional information regarding this training can be found here: http://eng.jhu.edu/wse/page/conduct-of-research-training.

- Failure to do so may result in a loss of funding. Additional information regarding this requirement can be found here: http://eng.jhu.edu/wse/page/conduct-of-research-training.

- A current listing of courses with area designators is provided on the departmental website.

- While this listing includes a few highly relevant courses outside the Department of Computer Science, only one such course may be applied toward the qualifying course requirement. Other than independent study courses, no courses with grades of P can be counted toward this Ph.D. qualifying course requirement.

**Tuition Support**

M.S.E. students are not normally eligible for tuition waivers, but will be able to work on campus up to 19.99 hours per week for hourly rates. There are also course assistant positions available for qualified students who are seeking financial support. Those interested must apply at the start of each semester for specific courses in need.

**Requirements for the Ph.D. Degree**

The goal of the Doctor of Philosophy (Ph.D.) program in the Department of Computer Science is to prepare first-rate scholars in the analysis, systems, and applications areas of computer science. Successful graduates may assume significant positions in academia, research institutes, industry, or government laboratories.

Applications for admission to the Ph.D. program in Computer Science are reviewed by a faculty committee. Although the specific criteria are not rigid, all students admitted will exhibit exceptional intellectual achievements and promise. Applicants must submit letters of recommendation, GRE scores, and (for foreign applicants) TOEFL scores.

In keeping with Hopkins’ traditions, program requirements are flexible, as described below. For more detailed policies regarding the PhD program, please visit the Advising Manual (http://www.cs.jhu.edu/graduate-studies/phd-requirements) on our departmental website.

**University Residency**

Two consecutive semesters of residence as a full-time graduate student are required.

**Seminar Attendance**

All Ph.D. degree candidates are required to enroll and maintain satisfactory attendance in Computer Science Seminar 600.601-602 each semester for the duration of their enrollment in the program. Although seminar attendance is required, the seminar may not be counted toward the qualifying course requirement.

**Responsible Conduct of Research**

All doctoral students are required to take AS.360.625 Responsible Conduct of Research. Students are expected to complete the course by the end of the first year but have until the start of the fourth semester to meet this requirement. Failure to do so may result in a loss of funding. Additional information regarding this requirement can be found here: http://eng.jhu.edu/wse/page/conduct-of-research-training.

**Qualifying Course Requirements**

The Department of Computer Science classifies its courses into three research areas: analysis, applications, and systems. All Ph.D. candidates must complete at least two graduate courses (3 class hours each) from each of these three areas. Each upper-level course description in this catalog includes its area for reference. A course in multiple areas may only be counted toward one requirement. A current listing of courses with area designators is provided on the departmental website. While this listing includes a few highly relevant courses outside the Department of Computer Science, only one such course may be applied toward the area requirements. Ph.D. students must also complete an additional two elective graduate courses (chosen from any CS area or from closely related departments such as Electrical and Computer Engineering, Cognitive Science, Mathematics, or Applied Mathematics and Statistics) for a total of eight graduate courses. In the Department of Computer Science, graduate courses are 600-level and above, or 400-level courses that are not offered at the 600-level. The coursework program must be approved by the student’s faculty advisor. The overall grade point average for these eight courses must be at least equivalent to a B+. No course with a grade of less than C- may be counted toward this Ph.D. qualifying course requirement. Other than independent study courses, no courses with grades of P can be counted toward the coursework requirement. Courses with grades of P will not be included in the grade point average calculation. One of the courses required for the degree, but only one, may be replaced by 3 credits from comparable short courses. With approval of the student’s faculty advisor, up to two courses can be transferred from graduate programs of other institutions; more than two such courses can be transferred.
with approval of the department. It is the obligation of the student to provide all necessary data to the Department of Computer Science regarding the course(s) for which transfer credit is being requested. Students are expected to complete the course requirements by the end of their second year as a Ph.D. candidate.

**Qualifying Project Requirements**

A Ph.D. student must complete two projects, each under the supervision and with the written agreement of a different faculty member in the Department of Computer Science. Upon conclusion of each project, the student must write a “Project Report” describing the project in detail. This report will be a public document and will be kept on file in the department office. The supervising faculty member must approve the project report. Students are expected to complete the qualifying projects by the end of their third year as a Ph.D. candidate.

Upon completion of the Ph.D. qualifying course requirements and the first qualifying project, students are ordinarily eligible to receive a master of science in engineering degree. The degree will be awarded upon student request.

**Graduate Board Oral Examination (GBO)**

This examination is a university requirement, ideally taken in the student’s third year. The oral exam is administered by a panel consisting of the research sponsor, two faculty members from the Department of Computer Science, and two from outside the department. The exam seeks to establish the student’s readiness to conduct original research in the area of his or her “Preliminary Research Proposal,” which should be distributed to the examiners in advance and presented by the student at the start of the exam.

**Part-Time Ph.D.**

Two consecutive semesters of residence as a full-time graduate student are required by the university. Part-time students must pass both the Ph.D. qualifying requirements and the Graduate Board oral exam within four years of being admitted to the program. Attempting to obtain a Ph.D. is a major commitment and involves close coordination with a faculty advisor in the department. Part-time students must be able to establish and maintain these close links.

**Departmental Seminar**

Ph.D. students must give an official departmental seminar on their research area. This is to be done after the GBO and prior to the dissertation defense, or as part of the dissertation defense.

**Dissertation and Defense**

Ph.D. students must write a dissertation consisting of original research in their chosen area. They must deliver a public presentation of the dissertation before a dissertation committee consisting of the faculty advisor, a second faculty member in the Department of Computer Science (who must have a primary tenure-track appointment in the Department if the advisor does not), and one or more other members with Ph.D. degrees. In conformity with University requirements, the members of the dissertation committee must submit a referee’s letter to the Graduate Board recommending that the dissertation be accepted. Completed dissertations will be formatted and submitted to the Milton S. Eisenhower Library for electronic publication (http://guides.library.jhu.edu/etd).

**Teaching Requirement**

Beginning in Fall 2015, all Ph.D. students are required to serve as a Teaching Assistant at least one semester during their program of study. As part of the requirement, the supervising course instructor must give the TA an opportunity to be in front of a group of students at least once during the course. Students are required to sign-up for the course EN.600.807 Teaching Practicum during the semester in which the requirement is being fulfilled, and at the end of the semester his/her performance will be evaluated by the course instructor.

**Student Progress Review**

Ph.D. students are reviewed annually by their advisor(s) and the department, and notified in writing as to their standing in the program. Students deemed not to be making satisfactory progress may be placed on probation.

**Financial Aid**

Financial aid is available for candidates of high promise. Fellowships provide a student with a stipend plus tuition. Research assistantships are available on sponsored research projects directed by members of the faculty. Teaching assistantships normally consist of tuition plus a stipend commensurate with the teaching or grading duties assigned. Students determined to have significant deficiency in spoken English may be required to take one or more semesters of English as a Second Language in order to qualify for employment as a teaching or research assistant.

For current faculty and contact information go to http://cs.jhu.edu/faculty/

**Faculty**

**Chair**

Yair Amir
Professor: distributed systems, resilient clouds and critical infrastructure, overlay networks, distributed algorithms.

**Professor Emeritus**

Gerald M. Masson
Fault tolerant computing, systems diagnosis theory, distributed computing, real-time error monitoring of hardware/software, software design for testability, interconnection networks.

**Professors**

Randal Burns
Storage systems, high performance and scientific computing, and database federations.

Jason M. Eisner
Computational linguistics (syntax and phonology), natural language processing, statistical machine learning, programming language design.

Gregory D. Hager
Vision, robotics, human-machine systems, computer-integrated medicine.

Philipp Koehn
Statistical machine translation.

S. Rao Kosaraju
Edward J. Schaefer Professor in Engineering: design of algorithms, pattern matching, and derandomization.

Nassir Navab
Augmented reality, vision, medical image computing and computer assisted interventions.

Aviel Rubin
Technical Director, Information Security Institute: system and networking security, computer privacy, applied cryptography.

Steven Salzberg
Bloomberg Distinguished Professor: genomics, bioinformatics, genome assembly, gene finding, sequence analysis algorithms.

Scott F. Smith
Programming languages, type systems, security in language design, component programming languages.

Alexander Szalay
Bloomberg Distinguished Professor: data intensive computing, theoretical astrophysics, galaxy formation.

Russell H. Taylor
John C. Malone Professor; Director, LCSR & CISST ERC: medical robotics, computer-integrated interventional medicine, medical image analysis, human-machine robotic systems.

David Yarowsky
Natural language and speech processing, information retrieval, machine translation, and machine learning.

Associate Professor
Michael Kazhdan
Computer graphics, surface reconstruction, image and geometry processing.

Assistant Professors
Yanif Ahmad
Data management, stream processing, declarative languages, parallel and distributed databases.

Raman Arora
Machine learning, statistical signal processing, stochastic approximation algorithms, applications to speech and language processing.

Alexis Battle
Genetics of complex traits, graphical models, transfer learning, structured regularization methods.

Vladimir Braverman
Algorithms, massive data sets, data streams, and database systems.

Michael Dinitz
Theoretical computer science, approximation algorithms, applications to networks and distributed computing.

Abhishek Jain
Cryptography, security, theoretical computer science.

Benjamin Langmead
Computational genomics, sequence alignment, text indexing, parallel and high performance computing.

Xin Li
Theory of computation, randomness, complexity theory, distributed computing, cryptography.

Suchi Saria
Machine learning, computational medicine, health informatics, and applications of machine learning in natural language processing, activity recognition and human-machine systems.

Research Professors
Amihood Amir
Algorithms design and analysis, multidimensional pattern matching, knowledge discovery algorithms, real time systems algorithms, computational molecular biology.

Bharat Doshi
Optical and wireless networking technologies, internet protocols and architectures, speech technologies and signal processing, and network design and analysis algorithms and tools.

Associate Research Professors
Giuseppe Ateniese
Applied cryptography, network security, and secure e-commerce.

Philippe Burlina
Computer vision, visual analysis and communications, multi-modality image exploitation, enterprise software systems for content and e-process management.

Chris Callison-Burch
Statistical natural language processing, machine translation, paraphrasing, evaluation of human language technologies.

Susan Hohenberger-Waters
Cryptography, computer security, algorithms, and complexity theory.

Peter Kazanzides
Medical robots, computer-assisted surgery, real-time systems.

James Mayfield
Information retrieval, cross-language retrieval, information extraction, natural language processing.

Andreas Terzis
P2P, overlay and sensor networks, resilient internet infrastructure, NP-based architectures.

Assistant Research Professors
Mark Dredze
Machine learning, natural language processing, health informatics.

Matthew Green
Applied cryptography, cryptographic protocol design, analysis of practical security systems, privacy-preserving storage and identification technologies.

Simon Leonard
Visual servoing, hand-eye coordination.

Christine Piatko
Computational geometry, information visualization, information retrieval.

Austin Reiter
Application of computer vision to robotics, machine vision, 3D reconstruction, image registration, visual recognition.

Benjamin Van Durme
Artificial intelligence, natural language processing (computational semantics), and streaming algorithms.

I-Jeng Wang
Wireless networking, Bayesian networks, probabilistic models.

Qingqing Zhang
Wireless communications and networking, Mobile Ad-hoc networks, cellular system and network technologies, multimedia applications and QoS, Internet protocol and algorithm design, performance analysis.

**Associate Teaching Professors**

Sara Miner More
Foundations of computing, computer science education, cryptography, information flow.

Joanne Selinski
Director of Undergraduate Studies: CS education, software engineering.

**Senior Lecturer**

Peter Fröhlich
Programming languages, software engineering, systems software, video game design, web applications.

**Part-time Lecturers**

Antonio DeSimone
Networks.

Sheela Kosaraju
Computer ethics.

Harold Lehmann
Medical informatics.

Christopher Pappacena
Cryptography.

**Visiting Professor**

Mitra Basu
Computational biology, pattern recognition, neural networks, artificial intelligence.

**Visiting Associate Professor**

H. Howie Huang
Computer systems and architecture, cloud computing, big data, high-performance computing & storage systems.

**Associate Research Scientist**

Anton Dahbura
Interim Executive Director, Information Security Institute: Information security, fault-tolerant computing, distributed systems, testing.

**Assistant Research Scientist**

Matt Post
Machine translation, syntax, parsing and language modeling.

**Adjunct Professors**

Gabor Fichtinger
Applied surgical robotics, surgical CAD/CAM systems, percutaneous therapies, stereotactic radiosurgery.

John W. Sheppard
Artificial intelligence, machine learning, data mining.

**Adjunct Assistant Professor**

Han Liu
Statistical machine learning, high dimensional nonparametric learning and massive-data analysis, multiple hypotheses testing, time series analysis, genomics, proteomics, cognitive neuroscience.

**Adjunct Associate Research Professors**

Seth Nielson
Network security.

Joshua Vogelstein
Scalable statistical inference methodologies, computational statistical methods.

Lanier Watkins
Critical infrastructure security, network security.

Lawrence B. Wolff
Computer vision, multi-sensor image fusion, augmented reality, biometrics.

**Joint Appointments**

Joel Bader
Associate Professor (Biomedical Engineering): bioinformatics and computational biology.

Emad Doctor
Assistant Professor (Radiology-Medical Imaging Physics): image-guided intervention, ultrasound imaging, elasticity, and thermal imaging.

Tamas Budavari
Assistant Professor (Applied Mathematics and Statistics): computational statistics, bayesian inference, low-dimensional embeddings, streaming and randomized algorithms.

Gregory Chirikjian
Professor (Mechanical Engineering): robotics, kinematics, dynamics, control, motion planning.

Noah Cowan
Associate Professor (Mechanical Engineering): sensor-based control of locomotion and manipulation, and biologically inspired robotics.

Ralph Etienne-Cummings
Professor (Electrical and Computer Engineering): mixed-signal VLSI, computational sensors, robotics, neuromorphic engineering.

James Fill
Professor (Applied Mathematics and Statistics): probability, stochastic processes, random structures, and algorithms.

Liliana Florea
Assistant Professor (McKusick-Nathans Institute for Genetic Medicine): application of computation techniques towards modeling and solving problems in biology and genetic medicine.

Rachel Karchin
Associate Professor (Biomedical Engineering): computational molecular biology, bioinformatics, genetic variation.

Sanjeev Khudanpur
Associate Professor (Electrical and Computer Engineering): information theory, statistical language modeling for speech recognition and machine translation

Elliot McVeigh
Professor (Biomedical Engineering): cardiovascular MRI, image guided therapy, novel MRI methods.

Michael I Miller
Professor (Biomedical Engineering): image understanding, computer vision, medical imaging, computational anatomy.

Mihaela Pertea
Assistant Professor (Medicine): computational tools for RNA-seq analysis, gene finding, splice site prediction, sequence motif finding.

Carey Priebe
Professor (Applied Mathematics and Statistics): computational statistics, kernel and mixture estimates, statistical pattern recognition, and statistical image analysis.

Jerry L. Prince
William B. Kouwenhoven Professor (Electrical and Computer Engineering) (Associate Director for Research, CISST ERC): image processing, computer vision, medical imaging.

Rebecca Schulman
Assistant Professor (Chemical and Biomolecular Engineering): molecular programming, DNA nanotechnology, self-assembly (theory and experiment), theoretical and systems biology, smart materials, nanoscale robotics.

Ralph Semmel
Professor (Director, APL): artificial intelligence, database systems, software engineering.

Jeff Siewerdsen
Professor (Biomedical Engineering): imaging physics, diagnostic radiology, image-guided interventions.

James Taylor
Associate Professor (Biology): genome informatics.

Rene Vidal
Associate Professor (Biomedical Engineering): computer vision, machine learning, robotics, and control.

Louis Whitcomb
Professor (Mechanical Engineering): dynamics and control of mechanical systems.

Raimond L. Winslow
Professor (Biomedical Engineering): modeling of biological systems, nonlinear systems theory, grid computing and data management, biomedical ontologies.

Thomas B. Woolf
Professor (Physiology): bioinformatics, computational biophysics.

For current course information and registration go to https://isis.jhu.edu/classes/

Courses

EN.600.101. MATLAB for Data Analytics.
MATLAB is a numerical-computing environment that is popular and powerful. In this course, you will learn to develop MATLAB scripts and functions to manipulate data, to visualize your results, and to share your results with others. Lecture time will be split between lecture and interactive sessions. This course is intended for students who are new to programming and numerical computing, though this is not a course on general-purpose programming.
Instructor(s): R. DiPietro
Area: Engineering.

Intended audience: students majoring in science, engineering or medicine. This course will provide a sampling of the theory behind and practical use of a broad spectrum of computational tools and technologies. We will start with Scratch (a programming language for kids) and show how many of the same concepts show up in Web Programming (HTML & Javascript). There will be a taste of algorithms, databases (SQL), Unix, statistics packages (R), data mining and visualization tools (graphviz), natural language processing, web search, interpreted languages (Python & LISP), compiled languages (C), and more. Students should come away with a few tools and concepts that will prove useful in their major, as well as the confidence that they can search the web to find what they need, when they need it, just-in-time.
Instructor(s): A. Irvine
Area: Engineering.

EN.600.104. Computer Ethics.
Students will examine a variety of topics regarding policy, legal, and moral issues related to the computer science profession itself and to the proliferation of computers in all aspects of society, especially in the era of the Internet. The course will cover various general issues related to ethical frameworks and apply those frameworks more specifically to the use of computers and the Internet. The topics will include privacy issues, computer crime, intellectual property law -- specifically copyright and patent issues, globalization, and ethical responsibilities for computer science professionals. Work in the course will consist of weekly assignments on one or more of the readings and a final paper on a topic chosen by the student and approved by the instructor. CS Majors Only - Alternate Weeks
Instructor(s): S. Kosaraju
Area: Humanities.

This course is required for all freshmen Computer Science majors. Transfers into the major and minors may enroll by permission only. Students will attend four 3-week blocks of meetings with different computer science professors, focused on a central theme. Active participation is required. Satisfactory/Unsatisfactory only.
Instructor(s): G. Hager; J. Selinski.
EN.600.107. Introductory Programming in Java.
This course introduces fundamental structured and object-oriented programming concepts and techniques, using Java, and is intended for all who plan to use computer programming in their studies and careers. Topics covered include variables, arithmetic operators, control structures, arrays, functions, recursion, dynamic memory allocation, files, class usage and class writing. Program design and testing are also covered, in addition to more advanced object-oriented concepts including inheritance and exceptions as time permits. First-time programmers are strongly advised to take EN.600.108 concurrently in Fall/Spring semesters.
Prerequisites: Students may receive credit for EN.600.107 or EN.600.112, but not both.
Instructor(s): S. More
Area: Engineering.

EN.600.108. Introduction to Programming Lab.
This course is intended for novice programmers, and must be taken in conjunction with EN.600.107. The purpose of this course is to give first-time programmers extra hands-on practice with guided supervision. Students will work in pairs each week to develop working programs, with checkpoints for each development phase. Must have familiarity with computers. Satisfactory/Unsatisfactory only.
Prerequisites: Coreq for EN.600.108: EN.600.107
Corequisites : Coreq: EN.600.107
Instructor(s): S. More
Area: Engineering.

EN.600.109. C++ Crash Course.
This is a crash course in C++ programming, primarily aimed at basic science students who need to use it as a tool in their field. The course will cover fundamental programming constructs in C++ including variables, decision statements and loops, along with input/output, functions, and arrays. Object oriented programming will be introduced if time permits. Students should have some prior programming experience in another language to include familiarity with variables, decisions statements and loops.
Instructor(s): O. Buzek
Area: Engineering.

EN.600.110. Python Scripting.
For non-majors, this is an introductory "learning by doing" course focused on the quick prototyping of computational solutions to problems from a variety of disciplines. After an introduction to the UNIX and IDLE environments we briefly cover the basics of programming in Python. We then spend the rest of the semester surveying a variety of powerful Python libraries, frameworks, and tools. We use these building blocks to create, for example, systems for image and sound processing, data analysis and visualization, event-based simulation, or database-driven web applications. There will be several sizeable team-based programming projects. [Note: This course may not be used for the CS major or minor requirements, except perhaps as a substitute for 600.107.]
Instructor(s): P. Froehlich
Area: Engineering.

EN.600.112. Introductory Programming for Scientists and Engineers.
An introductory "learning by doing" programming course for scientists, engineers, and everybody else who will need basic programming skills in their studies and careers. We cover the fundamentals of structured, modular, and (to some extent) object-oriented programming as well as important design principles and software development techniques such as unit testing and revision control. We will apply our shiny new programming skills by developing computational solutions to a number of real-world problems from a variety of disciplines. Students new to computer programming are encouraged to enroll into EN.600.113 IPSE Lab concurrently with this course. Students may receive credit for no more than one of the following: EN.600.107 or EN.600.111 or EN.600.112. [Note: This course may not be used for the CS major or minor requirements, except as a substitute for EN.600.107]
Prerequisites: Students who have taken EN.600.107 may not register for EN.600.112.;Students who have taken EN.600.111 may not register for EN.600.112.
Instructor(s): J. Selinski
Area: Engineering.

EN.600.113. IPSE Lab.
This course is intended for novice programmers, and must be taken in conjunction with EN.600.112. The purpose of this course is to give first-time programmers hands-on practice with guided supervision. Students may receive credit for EN.600.108 or EN.600.113, but not both. Co-requisite: EN.600.112. Satisfactory/Unsatisfactory only.
Prerequisites: Corequisite: EN.600.112;Students may receive credit for EN.600.108 or EN.600.113, but not both.
Instructor(s): J. Selinski
Area: Engineering.

EN.600.120. Intermediate Programming.
This course teaches intermediate to advanced programming, using C and C++ . (Prior knowledge of these languages is not expected.) We will cover low-level programming techniques, as well as object-oriented class design, and the use of class libraries. Specific topics include pointers, dynamic memory allocation, polymorphism, overloading, inheritance, templates, collections, exceptions, and others as time permits. Students are expected to learn syntax and some language specific features independently. Course work involves significant programming projects in both languages. Recommended Course Background: AP CS, EN.600.107, EN.600.110, EN.600.112 or equivalent.
Instructor(s): B. Langmead; S. More
Area: Engineering.

EN.600.127. Fun With Haskell.
Functional programming, which emphasizes the mathematical aspects of computations, is central to many modern efforts to build fast, scalable systems. This course will provide an introduction to the (lazy) functional style of computer programming, using the increasingly-popular Haskell language. Prerequisite: 600.107 or equivalent
Instructor(s): N. Filardo.
This hands-on course will be a gentle introduction to the field of embedded computing; computer systems that interact with the real world, making possible cell phones, MP3 players, flash drives, Wii games, and many more. Students will be programming different projects in modern microcontrollers, witnessing the effects immediately. The course will address the hardware-software interface of computer systems, setting a good foundation to understand the physical layer of computer applications and networking. The main topics are computer architecture fundamentals, assembly language, interfacing peripherals, programming device drivers, working with sensors, and data acquisition. Each student will be provided with the material to perform experiments individually, as well as with learning handouts. The course will be intensive but fun. (Projected syllabus http://www.cs.jhu.edu/~jorgev/cs333/ESF.pdf)
Area: Engineering.

EN.600.134. Embedded Software Architecture.
Most efforts to improve software quality focus exclusively on process improvements, such as adoption of coding standards and better use of static analysis tools, code reviews, and testing. But poor software quality also results from bad decisions about the architecture of that software. Software architecture is a critical area for all embedded programmers to understand because of the importance of meeting real-time deadlines and ensuring safe and reliable operation. This hands-on course teaches software architecture with a focus specifically on real-time and embedded software.
Prerequisites: Intermediate Programming EN.600.120 AND Computer System Fundamentals (EN.600.333 OR EN.600.433)
Instructor(s): M. Barr
Area: Engineering.

This course will give an introduction to the concepts and major elements of computer-integrated surgery (CIS) through clinical applications. Students will learn to ask questions and look for answers the way clinical engineers build and analyze CIS systems. Major topics will include medical imaging, image processing, surgical planning, surgical robotics, robot navigation, systems integration, and clinical validation. No computer programming will be necessary to complete the assignments.
Instructor(s): S. Leonard
Area: Engineering.

EN.600.146. Introduction to Medical Imaging.
This intersession class will provide an introduction to medical imaging. It will cover the following imaging modalities: X-ray, computed tomography (CT), ultrasound, photoacoustic, and magnetic resonance imaging (MRI). The basic principles, instrumentation, and applications of each imaging modality will be presented. The course will include a mixture of lectures, classroom discussions, student presentations, and imaging demos using medical imaging resources at Hackerman Hall. Assignments will test theoretical knowledge and practical applications. Introductory physics, chemistry, and pre-calculus math are recommended pre-requisites. Note: Students should not expect an in depth analysis of medical imaging systems. This class is not intended as a substitute for Medical Imaging courses offered during fall and spring terms.
Instructor(s): M. Bell
Area: Engineering.

EN.600.148. Engineering How Surgeons Learn to Operate on Patients.
Can the da Vinci robot sitting in Hackerman Hall operate with/like a surgeon? Can engineers change the world, “Millions at a time”? In this course, we will consider challenges surgeons face in learning how to operate. We will explore a few different techniques that have been developed for measuring surgical skill. Students will learn how to read and discuss scientific articles, and gain perspective on how engineering solutions developed in symbiosis with surgery and data and measurement sciences can help surgeons train to provide safe and effective patient care.
Area: Natural Sciences, Quantitative and Mathematical Sciences.

EN.600.211. Unix System Programming.
This course covers a variety of topics in UNIX programming, including process control, signal handling, daemon processes, and interprocess communication. Participants must be familiar with using the UNIX environment and be fluent in the C programming language.
Prerequisites: Prereq: EN.600.120
Instructor(s): P. Froehlich
Area: Engineering.

Prereq: 600.120 This course will provide students with a modern and historical understanding of malicious software (i.e., malware) from a technical and societal perspective. The class will consist of lectures, discussion, and programming projects in which students will identify, exploit, and correct vulnerabilities that have been leveraged by malware. Due to the nature of the class projects, this course is intended for students with programming experience.
Prerequisites: EN.600.120
Instructor(s): S. Small
Area: Engineering.

EN.600.221. An Introduction to Connectomics.
This course will introduce the emerging field of connectomics, and give students the opportunity to contribute directly to ongoing research efforts within the computer science department. This field enables novel brain circuit analysis at the ultrastructure level (i.e., individual synapses and neurons) and promises insight into areas such as biofidelic algorithms and the validation of the cortical column hypothesis first proposed at JHU by Vernon Mountcastle in the 1960s. We will begin by broadly surveying the field of brain mapping across different scales, and more deeply examine research in ultrastructure electron microscopy reconstruction efforts. Students will learn about scalable algorithms and approaches to extract graphs from large image volumes (O(100 TB+)), and the importance of computer science in addressing modern neuroscience challenges. Programming experience in MATLAB, Python or R is helpful but not required.
Instructor(s): W. Gray Roncal
Area: Engineering, Natural Sciences.

EN.600.226. Data Structures.
This course covers the design and implementation of data structures including collections, sequences, trees, and graphs. Other topics include sorting, searching, and hashing. Course work involves both written homework and Java programming assignments. Recommended Course Background: AP CS, EN.600.107 or EN.600.120
Instructor(s): G. Hager
Area: Engineering, Quantitative and Mathematical Sciences.
[Formerly EN.600.333/433] We study the design and performance of a variety of computer systems from simple 8-bit micro-controllers through 32/64-bit RISC architectures all the way to ubiquitous x86 CISC architecture. We'll start from logic gates and digital circuits before delving into arithmetic and logic units, registers, caches, memory, stacks and procedure calls, pipelined execution, super-scalar architectures, memory management units, etc. Along the way we'll study several typical instruction set architectures and review concepts such as interrupts, hardware and software exceptions, serial and other peripheral communications protocols, etc. A number of programming projects, frequently done in assembly language and using various processor simulators, round out the course. [Systems]. Students may receive credit for only one of EN.600.233, EN.600.333 or EN.600.433. Recommended Course Background: intro programming
Instructor(s): P. Froehlich
Area: Engineering.

EN.600.243. Introduction to Hardware Hacking.
Our favorite electronic devices, such as gaming consoles and smartphones, have a common root --- hardware. These deceptively simple interconnections of electronic components perform arithmetic and logic operations that enable our devices to interact with us and extend current methods for software security, communication, and marketing. In this course, we first survey hacks (e.g., rooting) that furnish a practical understanding of the hardware/software relationship. We continue this relationship by examining the use of hardware emulation and security extensions. Lastly, we explore other niche hardware uses such as asset tracking, advertising, and authentication. Recommended Background: Intermediate Programming, Data Structures, CSF.
Area: Engineering, Quantitative and Mathematical Sciences.

This course will give an introduction to the concepts and major elements of computer-integrated surgery (CIS) and its clinical applications. Major topics will include image processing, surgical planning, surgical robotics, surgical navigation, systems integration, and clinical validation. The class may include a clinical lab module. Grades will be calculated based on participation in class and three homework assignments. No computer programming will be necessary or required to complete the assignments. Recommended Course Background: Pre-calculus, Linear Algebra and Vector Calculus.
Instructor(s): R. Kumar
Area: Engineering.

EN.600.250. User Interfaces and Mobile Applications.
This course will provide students with a rich development experience, focused on the design and implementation of user interfaces and mobile applications. A brief overview of human computer interaction will provide context for designing, prototyping and evaluating user interfaces. Students will invent their own mobile applications and implement them using the Android SDK, which is JAVA based. An overview of the Android platform and available technologies will be provided, as well as XML for layouts, and general concepts for effective mobile development. Students will be expected to explore and experiment with outside resources in order to learn technical details independently. There will also be an emphasis on building teamwork skills, and on using modern development techniques and tools.
Prerequisites: EN.600.120 AND EN.600.226
Instructor(s): J. Selinski
Area: Engineering.

EN.600.255. Introduction to Video Game Design.
A broad survey course in video game design (as opposed to mathematical game theory), covering artistic, technical, as well as sociological aspects of video games. Students will learn about the history of video games, archetypal game styles, computer graphics and programming, user interface and interaction design, graphical design, spatial and object design, character animation, basic game physics, plot and character development, as well as psychological and sociological impact of games. Students will design and implement an experimental video game in interdisciplinary teams of 3-4 students as part of a semester-long project. Section 1 requires technical skills, including at least one programming course (preferably 2 or more). Section 2 requires artistic skills, including at least one multimedia course (preferably 2 or more). Open to sophomores and above.
Corequisites: Co-req: 600.256
Instructor(s): P. Froehlich
Area: Engineering.

EN.600.256. Introduction to Video Game Design Lab.
A lab course in support of 600.255: Introduction to Video Game Design covering a variety of multi-media techniques and applications from image processing, through sound design, to 3D modeling and animation. See 600.255: Introduction to Video Game Design for details about enrolling.
Corequisites: EN.600.255[C]
Instructor(s): P. Froehlich.

This course is an introduction to the theory of computing. topics include design of finite state automata, pushdown automata, linear bounded automata, Turing machines and phrase structure grammars; correspondence between automata and grammars; computable functions, decidable and undecidable problems, P and NP problems, NP-completeness, and randomization. Students may not receive credit for EN.600.271 and EN.600.471 for the same degree.
Prerequisites: EN.550.171
Instructor(s): X. Li
Area: Engineering, Quantitative and Mathematical Sciences.

EN.600.306. Introduction to Speech.
This course will introduce students to speech from an interdisciplinary perspective including Computer Science, Electrical Engineering, Linguistics and Psychology. Topics such as pitch will be discussed from a variety of perspectives including signal processing (estimating fundamental frequency), perception, linguistics and computational linguistics. Vowels will be described from multiple perspectives ranging from distinctive features in linguistics to formants in signal processing. Students will become familiar with a variety of topics ranging from spectrogram reading to using XML to program phones and Python (NLTK) to find interesting patterns in text corpora. To reach a diverse interdisciplinary audience, no background experience is required. Short course.
Instructor(s): K. Church
Area: Engineering.
EN.600.311. Sparse Representation in Computer Vision.
Techniques from sparse signal representation have seen significant impact in computer vision, often on non-traditional applications where the goal is not just to obtain a compact high-fidelity representation of the observed signal, but also to extract semantic information. The choice of dictionary plays a key role in bridging this gap: unconventional dictionaries consisting of, or learned from, the training samples themselves provide the key to obtaining state-of-the-art results and to attaching semantic meaning to sparse signal representations. Understanding the good performance of such unconventional dictionaries in turn demands new algorithmic and analytical techniques. This short course highlights a few representative examples of how the interaction between sparse signal representation and computer vision can enrich both fields. Recommended Course Background: EN.600.107, AS.110.201, and EN.550.310.
Area: Engineering.

EN.600.315. Databases.
Introduction to database management systems and database design, focusing on the relational and object-oriented data models, query languages and query optimization, transaction processing, parallel and distributed databases, recovery and security issues, commercial systems and case studies, heterogeneous and multimedia databases, and data mining. [Systems] (www.cs.jhu.edu/~yarowsky/cs415.html) Students may receive credit for EN.600.315 or EN.600.415, but not both.
Prerequisites: Prerequisite: EN.600.226
Instructor(s): D. Yarowsky
Area: Engineering.

EN.600.316. Database Systems.
This course serves as an introduction to the architecture and design of modern database management systems. Topics include query processing algorithms and data structures, data organization and storage, query optimization and cost modeling, transaction management and concurrency control, high-availability mechanisms, parallel and distributed databases, and a survey of modern architectures including NoSQL, column-oriented and streaming databases. Course work includes programming assignments and experimentation in a simple database framework written in Java. [Systems] Students may receive credit for EN.600.316 or EN.600.416, but not both.
Prerequisites: EN.600.120 AND EN.600.226; Students may receive credit for EN.600.316 or EN.600.416, but not both.
Instructor(s): Y. Ahmad
Area: Engineering.

This course covers fundamental topics related to operating systems theory and practice. Topics include processor management, storage management, concurrency control, multi-programming and processing, device drivers, operating system components (e.g., file system, kernel), modeling and performance measurement, protection and security, and recent innovations in operating system structure. Course work includes the implementation of operating systems techniques and routines, and critical parts of a small but functional operating system. Students may receive credit for EN.600.318 or EN.600.418 but not both. Recommended Course Background: EN.600.211
Prerequisites: Students may receive credit for EN.600.318 or EN.600.418 but not both. Prereqs: EN.600.120 AND EN.600.226 AND (EN.600.233 OR EN.600.333)
Instructor(s): P. Froehlich
Area: Engineering.

EN.600.320. Parallel Programming.
This course prepares the programmer to tackle the massive data sets and huge problem size of modern scientific and enterprise computing. Google and IBM have commented that undergraduate CS majors are unable to "break the single server mindset" (http://www.google.com/intl/en/press/pressrel/20071008_ibm_univ.html). Students taking this course will abandon the comfort of serial algorithmic thinking and learn to harness the power of cutting-edge software and hardware technologies. The issue of parallelism spans many architectural levels. Even `"single server" systems must parallelize computation in order to exploit the inherent parallelism of recent multi-core processors. The course will examine different forms of parallelism in four sections. These are: (1) massive data-parallel computations with Hadoop; (2) programming compute clusters with MPI; (3) thread-level parallelism in Java; and, (4) GPGPU parallel programming with NVIDIA’s Cuda. Each section will be approximately 3 weeks and each section will involve a programming project. The course is also suitable for undergraduate and graduate students from other science and engineering disciplines that have prior programming experience. [Systems] Students may receive credit for EN.600.320 or EN.600.420, but not both. Recommended Course Background: EN.600.233
Prerequisites: Prereq: EN.600.120 and EN.600.226; recommended 600.333/433
Instructor(s): R. Burns
Area: Engineering.

EN.600.321. Object Oriented Software Engineering.
This course covers object-oriented software construction methodologies and their application. The main component of the course is a large team project on a topic of your choosing. Course topics covered include object-oriented analysis and design, UML, design patterns, refactoring, program testing, code repositories, team programming, and code reviews. [Systems or Applications] (http://pl.cs.jhu.edu/~yarowsky/oose/index.shtml) Students may receive credit for EN.600.321 or EN.600.421, but not both.
Prerequisites: EN.600.120 AND EN.600.226
Instructor(s): S. Smith
Area: Engineering.

Data-Intensive Computing is an experiential education course in computing with massive data sets that covers the software, algorithms, and systems used to ingest, store, and analyze. Specific topics include: NoSQL software systems including key/value stores and graph databases, scientific python, array databases, (semi)-external memory array and graph algorithms, extract-transform-load (ETL) processing, spatial indexing, OpenCL GPU code acceleration, and performance management of clusters. The course will utilize the unique computing resources at JHU, including the DataScope (5PB of storage), the GPU cluster (110 TFlops), and the Homewood High Performance Computing Cluster (1600 cores). The entire course will take place in several lengthy lab sessions each week. Course time will be divided roughly into team projects (30%), ad-hoc tasks (50%), presentation (10%) and using collaboration tools for concurrent reading and authoring and interactive self-assessment. [Systems]
Prerequisites: EN.600.320 OR EN.600.420
Instructor(s): R. Burns
Area: Engineering.
EN.600.325. Declarative Methods.
Suppose you could simply write down a description of your problem, and let the computer figure out how to solve it. What notation could you use? What strategy should the computer then use? In this survey class, you'll learn to recognize when your problem is an instance of satisfiability, constraint programming, logic programming, dynamic programming, or mathematical programming (e.g., integer linear programming). For each of these related paradigms, you'll learn to reformulate hard problems in the required notation and apply off-the-shelf software that can solve any problem in that notation -- including NP-complete problems and many of the problems you'll see in other courses and in the real world. You'll also gain some understanding of the general-purpose algorithms that power the software. [Analysis] Students can only receive credit for EN.600.325 or EN.600.425, not both.
Prerequisites: EN.600.226
Instructor(s): J. Eisner
Area: Engineering.

EN.600.328. Compilers and Interpreters.
Introduction to compiler design, including lexical analysis, parsing, syntax-directed translation, symbol tables, run-time environments, and code generation and optimization. Students are required to write a compiler as a course project. [Systems] Co-listed with EN.600.428
Prerequisites: EN.600.120 AND EN.600.226
Instructor(s): P. Froehlich
Area: Engineering.

EN.600.335. Artificial Intelligence.
The course situates the study of Artificial Intelligence (AI) first in the broader context of Philosophy of Mind and Cognitive Psychology and then treats in-depth methods for automated reasoning, automatic problem solvers and planners, knowledge representation mechanisms, game playing, machine learning, and statistical pattern recognition. The class is a recommended for all scientists and engineers with a genuine curiosity about the fundamental obstacles to getting machines to perform tasks such as deduction, learning, and planning and navigation. Strong programming skills and a good grasp of the English language are expected; students will be asked to complete both programming assignments and writing assignments. The course will include a brief introduction to scientific writing and experimental design, including assignments to apply these concepts. [Applications] Prereq: 600.226, 550.171; Recommended: linear algebra, prob/stat. Students can only receive credit for 600.335 or 600.435, not both.
Prerequisites: EN.600.226[C] AND EN.550.171[C]
Instructor(s): P. Koehn
Area: Engineering.

This is an introductory course presenting a series of algorithms related to the representation and use of geometric models acquired from sensor data. Course topics include: basic sensing and estimation techniques, geometric model representations, and motion planning algorithms. The course will also discuss applications in diverse areas such as mobile systems, robot manipulation, and medicine. Students may receive credit for EN.600.336 or EN.600.436, but not both.
Recommended Course Background: AS.110.106, probability/statistics.
Prerequisites: EN.600.226[C]
Instructor(s): G. Hager
Area: Engineering.

EN.600.337. Distributed Systems.
This course teaches how to design and implement protocols that enable processes to exchange information, cooperate, and coordinate efficiently in a consistent manner over a computer network. Topics include communication protocols, group communication, distributed databases, distributed operating systems, and security. [Systems] Students may receive credit for EN.600.337 or EN.600.437 but not both.
Instructor(s): Y. Amir
Area: Engineering.

EN.600.340. Introduction to Genomic Research.
This course will use a project-based approach to introduce undergraduates to research in computational biology and genomics. During the semester, students will take a series of large data sets, all derived from recent research, and learn all the computational steps required to convert raw data into a polished analysis. Data challenges might include the DNA sequences from a bacterial genome project, the RNA sequences from an experiment to measure gene expression, the DNA from a human microbiome sequencing experiment, and others. Topics may vary from year to year. In addition to computational data analysis, students will learn to do critical reading of the scientific literature by reading high-profile research papers that generated groundbreaking or controversial results. [Applications] Recommended Course Background: Knowledge of the Unix operating system and programming expertise in a language such as Perl or Python.
Instructor(s): S. Salzberg
Area: Engineering.

Topics covered will include application layer protocols (e.g. HTTP, FTP, SMTP), transport layer protocols (UDP, TCP), network layer protocols (e.g. IP, ICMP), link layer protocols (e.g. Ethernet) and wireless protocols (e.g. IEEE 802.11). The course will also cover routing protocols such as link state and distance vector, multicast routing, and path vector protocols (e.g. BGP). The class will examine security issues such as firewalls and denial of service attacks. We will also study DNS, NAT, Web caching and CDNs, peer to peer, and protocol tunneling. Finally, we will explore security protocols (e.g. TLS, SSH, Psec), as well as some basic cryptography necessary to understand these. Grading will be based on hands-on programming assignments, homeworks and two exams. [Systems] Students can only receive credit for EN.600.344 or EN.600.444, not both.
Prerequisites: Prereqs For EN.600.344: EN.600.233/333/433 or permission. Students can only receive credit for EN.600.344 or EN.600.444, not both
Instructor(s): A. DeSimone
Area: Engineering.

EN.600.355. Video Game Design Project.
An intensive capstone design project experience in video game development. Students will work in groups of 4-8 on developing a complete video game of publishable quality. Teams will (hopefully) include programmers, visual artists, composers, and writers. Students will be mentored by experts from industry and academia. Aside from the project itself, project management and communication skills will be emphasized. Enrollment is limited to ensure parity between the various disciplines. [General] May involve travel to MICA. Junior or senior standing recommended.
Prerequisites: EN.600.255 AND EN.600.256 or permission of instructor
Instructor(s): P. Froehlich
Area: Engineering.
This course introduces computer graphics techniques and applications, including image processing, rendering, modeling and animation. [Applications] Students may receive credit for EN.600.357 or EN.600.457, but not both. No Audits.
**Prerequisites:** EN.600.120 AND EN.600.226
Instructor(s): M. Kazhdan
Area: Engineering, Quantitative and Mathematical Sciences.

EN.600.363. Introduction To Algorithms.
This course concentrates on the design of algorithms and the rigorous analysis of their efficiency. Topics include the basic definitions of algorithmic complexity (worst case, average case); basic tools such as dynamic programming, sorting, searching, and selection; advanced data structures and their applications (such as union-find); graph algorithms and searching techniques such as minimum spanning trees, depth-first search, shortest paths, design of online algorithms and competitive analysis. [Analysis] Students may receive credit for EN.600.363 or EN.600.463, but not both.
**Prerequisites:** EN.600.226 and EN.550.171 or permission
Instructor(s): V. Braverman
Area: Engineering, Quantitative and Mathematical Sciences.

EN.600.371. Software Tools Practicum.
This course will survey the wide range of tools, frameworks and packages that form a foundation for much of the modern Internet. Each week, a group of students will present a particular software package to the class, exploring the architectural, historical and competitive aspects. Students will work with others to explore the packages, learn how they work and then summarize this knowledge by creating a presentation for the class. [General] CS majors only.
**Prerequisites:** EN.600.120 AND EN.600.226
Instructor(s): P. Wayner
Area: Engineering.

EN.600.384. Augmented Reality.
Undergraduate level version of EN.600.684. Students may take EN.600.384 or EN.600.684, but not both.
**Prerequisites:** Students may take EN.600.384 or EN.600.684, but not both.;EN.600.120 AND EN.600.226AND AS.110.201
Linear Algebra
Instructor(s): N. Navab
Area: Engineering.

EN.600.392. CS Design Project.
This course will give junior and senior CS majors an intensive design project experience. Students will work in groups with real world customers to develop a working system. Project design, management and communication skills will be emphasized. Software development methodologies may also be presented. Recommended Course Background: EN.600.321
**Prerequisites:** EN.600.120 AND EN.600.226; Recommended 600.321
Instructor(s): P. Froehlich
Area: Engineering.

EN.600.402. Digital Health and Biomedical Informatics.
Advances in technology are driving a change in medicine, from personalized medicine to population health. Computers and information technology will be critical to this transition. We shall discuss some of the coming changes in terms of computer technology, including computer-based patient records, clinical practice guidelines, and region-wide health information exchanges. We will discuss the underlying technologies driving these developments - databases and warehouses, controlled vocabularies, and decision support.
Instructor(s): H. Lehmann
Area: Engineering.

EN.600.411. Computer Science Innovation & Entrepreneurship II.
This course is the second half of a two-course sequence and is a continuation of course EN.600.410.01, CS Innovation and Entrepreneurship, offered by the Center for Leadership Education (CLE). In this sequel course the student groups, directed by CS faculty, will implement the business idea which was developed in the first course and will present the implementations and business plans to an outside panel made up of practitioners, industry representatives, and venture capitalists. [General]
**Prerequisites:** EN.600.410 AND ( EN.600.321 OR EN.600.421)
Instructor(s): A. Dahbura; L. Aronhime
Area: Engineering.

EN.600.415. Databases.
Graduate level version of EN.600.315 [Systems]. Students may receive credit for EN.600.315 or EN.600.415, but not both. Recommended Course Background: EN.600.226
Instructor(s): D. Yarowsky
Area: Engineering.

Similar material as EN.600.316, covered in more depth. Intended for upper-level undergraduates and graduate students. Students may receive credit for EN.600.316 or EN.600.416, but not both. Recommended Course Background: EN.600.120 and EN.600.226
**Prerequisites:** Students may receive credit for EN.600.316 or EN.600.416, but not both.
Instructor(s): Y. Ahmad
Area: Engineering.

EN.600.418. Operating Systems.
Similar material as EN.600.318, covered in more depth. Intended for upper-level undergraduates and graduate students. Students may receive credit for EN.600.318 or EN.600.418, but not both. [Systems]
**Prerequisites:** Students may receive credit for EN.600.318 or EN.600.418 but not both.
Instructor(s): P. Froehlich
Area: Engineering.

EN.600.420. Parallel Programming.
Graduate level version of EN.600.320. Students may receive credit for EN.600.320 or EN.600.420, but not both. Recommended Course Background: EN.600.120 or equivalent.
Instructor(s): R. Burns
Area: Engineering.

EN.600.421. Object Oriented Software Engineering.
Graduate level version of EN.600.321 [Systems or Applications].
Students may receive credit for EN.600.321 or EN.600.421, but not both. Recommended Course Background: EN.600.226 and EN.600.120
Instructor(s): S. Smith
Area: Engineering.
EN.600.423. **Data-Intensive Computing.**
Graduate student version of EN.600.323. [Systems] Students may receive credit for EN.600.323 or EN.600.423, but not both.
**Prerequisites: EN.600.320 OR EN.600.420**
Instructor(s): R. Burns
Area: Engineering.

EN.600.424. **Network Security.**
This course focuses on communication security in computer systems and networks. The course is intended to provide students with an introduction to the field of network security. The course covers network security services such as authentication and access control, integrity and confidentiality of data, firewalls and related technologies, Web security and privacy. Course work involves implementing various security techniques. A course project is required. [Systems] EN.600.120 (or equivalent) recommended. Recommend Course Background: 600.120, 600.226, 600.344, 600.444 or permission.
**Prerequisites: 600.226 and (600.344 or 600.444) or permission; 600.120 (or equivalent) recommended.**
Instructor(s): S. Nielson
Area: Engineering.

EN.600.425. **Declarative Methods.**
Students can only receive credit for EN.600.325 or EN.600.425, not both. Graduate level version of EN.600.325. Recommended Course Background: EN.600.226, EN.600.271, AS.110.107/AS.110.109
Instructor(s): J. Eisner
Area: Engineering.

EN.600.426. **Principles of Programming Languages.**
Functional, object-oriented, and other language features are studied independent of a particular programming language. Students become familiar with these features by implementing them. Most of the implementations are in the form of small language interpreters. Some type checkers and a small compiler will also be written. The total amount of code written will not be overly large, as the emphasis is on concepts. The ML programming language is the implementation language used. [Analysis] Requisites include 600.226. No Freshmen or Sophomores.
Instructor(s): S. Smith
Area: Engineering, Quantitative and Mathematical Sciences.

EN.600.428. **Compilers & Interpreters.**
Introduction to compiler design, including lexical analysis, parsing, syntax-directed translation, symbol tables, run-time environments, and code generation and optimization. Students are required to write a compiler as a course project. Co-listed with EN.600.328. Students should have knowledge of C/C++ programming and data structures. Graduate version of EN.600.328. Students may receive credit for EN.600.328 or EN.600.428, but not both.
**Prerequisites: EN.600.120 AND EN.600.226**
Instructor(s): P. Froehlich
Area: Engineering.

EN.600.429. **Functional Programming at Work - Haskell and Domain-Specific Languages.**
This course studies pure functional programming in the Haskell language and the use of functional programming to build domain specific languages (DSLs): customized, application specific programming languages. This course starts with an introduction to Haskell and its essential ideas of lazy evaluation and type inference. Advanced functional programming topics will include type classes, monads and monad transformers, arrows, templates, dependent types, parser combinators, and multiple parameter type classes. The class will study existing DSLs and DSL implementation techniques, including languages for reactive programming, computer vision, hardware design, computer music, and parallel processing. Students will implement a DSL of their choice in Haskell.
Instructor(s): J. Peterson
Area: Engineering.

EN.600.430. **Ontologies and Knowledge Representation.**
Knowledge representation (KR) deals with the possible structures by which the content of what is known can be formally represented in such a way that queries can be posed and inferences drawn. Ontology concerns the hierarchical classification of entities from given domains of knowledge together with the relations between various classes or subclasses. We begin with KR, examining the standard variety of frameworks developed or implemented over the last twenty years, including 1st-order logic and automated theorem proving, networks, frames, and description logics. Then we move on to a study of the problems inherent in ontology development and examine the some of the currently prevalent environments, including Universal Modeling Language, OWL and Protege’, RDFS and semantic web applications. [Analysis] Recommended Course Background: EN.600.107 and EN.600.271
Instructor(s): R. Rynasiewicz
Area: Humanities, Quantitative and Mathematical Sciences.

EN.600.433. **Computer Systems.**
Graduate version of 600.333. Students may receive credit for 600.333 or 600.433, but not both. [Systems]
Instructor(s): P. Froehlich
Area: Engineering.

EN.600.435. **Artificial Intelligence.**
Students may receive credit for EN.600.335 or EN.600.435, not both. Graduate level version of EN.600.335 [Applications]. Prerequisite: EN.600.226, EN.550.171; Recommended: linear algebra, prob/stat.
Instructor(s): P. Koehn
Area: Engineering.

EN.600.436. **Algorithms for Sensor-Based Robotics.**
This course surveys the development of robotic systems for navigating in an environment from an algorithmic perspective. It will cover basic kinematics, configuration space concepts, motion planning, and localization and mapping. It will describe these concepts in the context of the ROS software system, and will present examples relevant to mobile platforms, manipulation, robotics surgery, and human-machine systems. [Analysis] Formerly EN.600.336. Students may receive credit for only one of EN.600.336, EN.600.436 and EN.600.636.
**Prerequisites: EN.600.226 and Linear Algebra and Probability; Students may receive credit for only one of EN.600.336, EN.600.436 and EN.600.636.**
Instructor(s): S. Leonard
Area: Engineering.
EN.600.437. Distributed Systems.
Graduate version of 600.337 Systems. Students may receive credit for 600.337 or 600.437 but not both. Recommended Course Background: EN.600.120, EN.600.226
Instructor(s): Y. Amir
Area: Engineering.

EN.600.438. Computational Genomics: Data Analysis.
Genomic data has the potential to reveal causes of disease, novel drug targets, and relationships among genes and pathways in our cells. However, identifying meaningful patterns from high-dimensional genomic data has required development of new computational tools. This course will cover current approaches in computational analysis of genomic data with a focus on statistical methods and machine learning. Topics will include disease association, prediction tasks, clustering and dimensionality reduction, data integration, and network reconstruction. There will be some programming and a project component. [Applications] Recommended Course Background: EN.600.226 or other programming experience, probability and statistics, linear algebra or calculus. Students may receive credit for EN.600.438 or EN.600.638, but not both.
Prerequisites: Students may receive credit for EN.600.438 or EN.600.638, but not both.]
Instructor(s): A. Battle
Area: Engineering.

EN.600.439. Computational Genomics.
Your genome is the blueprint for the molecules in your body. It’s also a string of letters (A, C, G and T) about 3 billion letters long. How does this string give rise to you? Your heart, your brain, your health? This, broadly speaking, is what genomics research is about. This course will familiarize you with a breadth of topics from the field of computational genomics. The emphasis is on current research problems, real-world genomics data, and efficient software implementations for analyzing data. Topics will include: string matching, sequence alignment and indexing, assembly, and sequence models. Course will involve significant programming projects. [Applications]
Prerequisites: EN.600.120 AND EN.600.226
Instructor(s): B. Langmead
Area: Engineering.

Genomic data is becoming available in large quantities, but understanding how genetics contributes to human disease and other traits remains a major challenge. Machine learning approaches allow us to automatically analyze and combine genomic data, build predictive models, and identify genetic elements important to disease and cellular processes. This course will cover uses of machine learning in diverse genomic applications. Students will present and discuss current literature. Topics include predicting disease risk from genomic data, integrating diverse genomic data types, gene network reconstruction, and other topics guided by student interest. The course will include a project component with the opportunity to explore publicly available genomic data. Recommended course background: coursework in data mining, machine learning. [Applications] Students may receive credit for 600.441 or 600.641, but not both.
Prerequisites: Students may receive credit for 600.441 or 600.641, but not both.
Instructor(s): A. Battle
Area: Engineering.

EN.600.442. Modern Cryptography.
This course focuses on cryptographic algorithms, formal definitions, hardness assumptions, and proofs of security. Topics include number-theoretic problems, pseudo-randomness, block and stream ciphers, public-key cryptography, message authentication codes, and digital signatures. Recommended Course Background: EN.600.226 and a 300-level or above systems course; EN.600.271/EN.600.471 and EN.550.171 or equivalent.
Instructor(s): A. Jain
Area: Engineering, Quantitative and Mathematical Sciences.

Lecture topics will include computer security, network security, basic cryptography, system design methodology, and privacy. There will be a heavy work load, including written homework, programming assignments, exams and a comprehensive final. The class will also include a semester-long project that will be done in teams and will include a presentation by each group to the class. [Applications]
Recommended Course Background: A basic course in operating systems and networking, or permission of instructor.
Instructor(s): M. Green
Area: Engineering.

EN.600.444. Computer Networks.
This course considers intersystem communications issues. Topics covered include layered network architectures; the OSI model; bandwidth, data rates, modems, multiplexing, error detection/correction; switching; queuing models, circuit switching, packet switching; performance analysis of protocols, local area networks; and congestion control. Recommended Course Background: EN.600.120 and EN.600.233. Students can only receive credit for EN.600.344 or EN.600.444, not both.
Prerequisites: Students can only receive credit for EN.600.344 or EN.600.444, not both.
Instructor(s): A. Rubin
Area: Engineering.

This course focuses on computer-based techniques, systems, and applications exploiting quantitative information from medical images and sensors to assist clinicians in all phases of treatment from diagnosis to preoperative planning, execution, and follow-up. It emphasizes the relationship between problem definition, computer-based technology, and clinical application and includes a number of guest lectures given by surgeons and other experts on requirements and opportunities in particular clinical areas. Required Course Background: AS.110.201 or permission of instructor. Recommended Course Background: EN.600.120, EN.600.457, EN.600.461, image processing.
Prerequisites: EN.600.226
Instructor(s): R. Taylor
Area: Engineering.
EN.600.446. Computer Integrated Surgery II.
This weekly lecture/seminar course addresses similar material to EN.600.445, but covers selected topics in greater depth. In addition to material covered in lectures/seminars by the instructor and other faculty, students are expected to read and provide critical analysis/presentations of selected papers in recitation sessions. Students taking this course are required to undertake and report on a significant term project under the supervision of the instructor and clinical end users. Typically, this project is an extension of the term project from EN.600.445, although it does not have to be. Grades are based both on the project and on classroom recitations. Students wishing to attend the weekly lectures as a 1-credit seminar should sign up for EN.600.452. Students may also take this course as EN.600.646. The only difference between EN.600.446 and EN.600.646 is the level of project undertaken. Typically, EN.600.646 projects require a greater degree of mathematical, image processing, or modeling background. Prospective students should consult with the instructor as to which course number is appropriate. [Applications] Students may receive credit for EN.600.446 or EN.600.646, but not both.

Prerequisites: Prereq for EN.600.446: EN.600.445 or EN.600.645 or permission
Instructor(s): R. Taylor
Area: Engineering.

This course is an introduction to fundamental concepts of networked embedded systems and wireless sensor networks. It is intended for juniors, seniors and first year graduate students in Computer Science and other engineering majors with the prerequisite background. Covered topics include: embedded systems programming concepts, low power and power aware design, radio technologies, communication protocols for ubiquitous computing systems, and some of the mathematical foundation of sensor behavior. Laboratory work consists of a set of programming assignments that consider a set of the issues described in class. Recommended Course Background: EN.600.226, EN.600.120, and EN.600.344/EN.600.444
Instructor(s): M. Chang
Area: Engineering.

EN.600.451. Introduction to Bitcoin and Other Cryptocurrencies.
This course covers the basics of Bitcoin and the underlying technologies driving it. The course is intended for students interested in the cryptographic techniques devised to make digital currencies and payment systems secure. Topics include Bitcoin transactions, the blockchain, mining, and decentralized consensus. The course will include a brief introduction to public-key cryptography, digital signatures, hash functions, proof of work/space, multisignatures, and elliptic curve cryptography. The course concludes with an overview of the Bitcoin scripting language and Bitcoin 2.0 platforms. [Systems] Recommended Course Background: EN.600.344/444 (Computer Networks) and EN.550.171 (Discrete Math)
Prerequisites: EN.600.226
Instructor(s): G. Ateniese
Area: Engineering.

EN.600.452. Seminar: Computer Integrated Surgery II.
Students may receive credit for EN.600.446 or EN.600.452, but not both. Lecture only version of EN.600.446 (no project). Recommended Course Background: EN.600.445 or instructor permission required.
Instructor(s): R. Taylor
Area: Engineering.

EN.600.454. Practical Cryptographic Systems.
This semester-long course will teach systems and cryptographic design principles by example: by studying and identifying flaws in widely-deployed cryptographic products and protocols. Our focus will be on the techniques used in practical security systems, the mistakes that lead to failure, and the approaches that might have avoided the problem. We will place a particular emphasis on the techniques of provable security and the feasibility of reverse-engineering undocumented cryptographic systems. [Systems]
Instructor(s): M. Green
Area: Engineering.

Graduate level version of EN.600.357. Students may receive credit for EN.600.357 or EN.600.457, but not both. Recommended Course Background: EN.600.120, EN.600.226, AS.110.201 or instructor permission.
Instructor(s): M. Kazhdan
Area: Engineering, Quantitative and Mathematical Sciences.

EN.600.459. Computational Geometry.
This course will provide an introduction to computational geometry. It will cover a number of topics in two- and three-dimensions, including polygon triangulations and partitions, convex hulls, Delaunay and Voronoi diagrams, arrangements, and spatial queries. Time-permitting, we will also look at kd-trees, general BSP-trees, and quadtrees. [Analysis] Recommended Course Background: EN.600.120 AND EN.600.226 AND (EN.600.363 OR EN.600.463).
Prerequisites: Students may receive credit for EN.600.459 or EN.600.659, but not both.
Instructor(s): M. Kazhdan
Area: Engineering, Quantitative and Mathematical Sciences.

EN.600.460. Software Vulnerability Analysis.
This course will examine vulnerabilities in C source, stack overflows, writing shell code, etc. Also, vulnerabilities in web applications: SQL Injection, cookies, as well as vulnerabilities in C binary fuzzing, and exploit development without source among other topics. Co-listed with EN.650.460
Instructor(s): S. Checkoway
Area: Engineering.

EN.600.461. Computer Vision.
This course gives an overview of fundamental methods in computer vision from a computational perspective. Methods studied include: camera systems and their modelling, computation of 3-D geometry from binocular stereo, motion, and photometric stereo; and object recognition. Edge detection and color perception are covered as well. Elements of machine vision and biological vision are also included. Students may receive credit for at most one of EN.600.361 or EN.600.461 or EN.600.661. [Applications] Prerequisites (soft): intro programming, linear algebra, and prob/stat.
Prerequisites: If you have completed EN.600.361 OR EN.600.661 you cannot enroll in EN.600.461.
Instructor(s): A. Reiter
Area: Engineering, Quantitative and Mathematical Sciences.

EN.600.463. Algorithms I.
Graduate version of EN.600.363. Students may receive credit for EN.600.363 or EN.600.463, but not both. Recommended Course Background: EN.600.226 and EN.550.171 or instructor permission required.
Instructor(s): V. Braverman
Area: Engineering, Quantitative and Mathematical Sciences.
EN.600.464. Randomized and Big Data Algorithms.
The course emphasizes algorithmic design aspects, and how randomization can be a helpful tool. The topics covered include: tail inequalities, linear programming relaxation & randomized rounding, de-randomization, existence proofs, universal hashing, markov chains, metropolis and metropolis-hastings methods, mixing by coupling and by eigenvalues, counting problems, semi-definite programming and rounding, lower bound arguments, and applications of expanders.
[Analysis] (www.cs.jhu.edu/~cs464) Recommended Course Background: EN.600.363 or EN.600.463.
Instructor(s): V. Braverman
Area: Engineering, Quantitative and Mathematical Sciences.

EN.600.465. Natural Language Processing.
This course is an in-depth overview of techniques for processing human language. How should linguistic structure and meaning be represented? What algorithms can recover them from text? And crucially, how can we build statistical models to choose among the many legal answers? The course covers methods for trees (parsing and semantic interpretation), sequences (finite-state transduction such as morphology), and words (sense and phrase induction), with applications to practical engineering tasks such as information retrieval and extraction, text classification, part-of-speech tagging, speech recognition and machine translation. There are a number of structured but challenging programming assignments. [Applications] Recommended Course Background: EN.600.226
Instructor(s): J. Eisner
Area: Engineering.

EN.600.466. Information Retrieval and Web Agents.
An in-depth, hands-on study of current information retrieval techniques and their application to developing intelligent WWW agents. Topics include a comprehensive study of current document retrieval models, mail/news routing and filtering, document clustering, automatic indexing, query expansion, relevance feedback, user modeling, information visualization and usage pattern analysis. In addition, the course explores the range of additional language processing steps useful for template filling and information extraction from retrieved documents, focusing on recent, primarily statistical methods. The course concludes with a study of current issues in information retrieval and data mining on the World Wide Web. Topics include web robots, spiders, agents and search engines, exploring both their practical implementation and the economic and legal issues surrounding their use. Recommended Course Background: EN.600.226
Instructor(s): D. Yarowsky
Area: Engineering.

EN.600.467. Wireless Networks.
This course covers the basics of mobile communication and wireless networking for computer science majors by keeping a balance between communication and networking topics. In this course the students will be exposed to wireless transmission fundamentals (path loss, shadowing, modulation, coding and channel models), wireless cellular networks (cellular concept, channel reuse, capacity limits, and cellular systems such as GSM, GPRS and UMTS), and learn about mobile network and transport layers, medium access control protocols, wireless local area networks (IEEE 802.11), wireless mesh networks (IEEE 802.16), and emerging dynamic spectrum access networks based on cognitive radios. Recommended Course Background: EN.600.344/EN.600.444 or equivalent.
Prerequisites: EN.600.344 OR EN.600.444 or equivalent
Instructor(s): A. Mishra
Area: Engineering.

Google translate can instantly translate between any pair of over fifty human languages (for instance, from French to English). How does it do that? Why does it make the errors that it does? And how can you build something better? Modern translation systems learn to translate by reading millions of words of already translated text, and this course will show you how they work. The course covers a diverse set of fundamental building blocks from linguistics, machine learning, algorithms, data structures, and formal language theory, along with their application to a real and difficult problem in artificial intelligence. Recommended Course Background: prob/stat, EN.600.226; EN.600.465
Instructor(s): P. Koehn
Area: Engineering.

EN.600.469. Approximation Algorithms.
This course provides an introduction to approximation algorithms. Topics include vertex cover, TSP, Steiner trees, cuts, greedy approach, linear and semi-definite programming, primal-dual method, and randomization. Additional topics will be covered as time permits. There will be a final project. Students may receive credit for EN.600.469 or EN.600.669, but not both. [Analysis]
Prerequisites: EN.600.363 OR EN.600.463 OR permission of instructor.
Instructor(s): M. Dinitz
Area: Engineering, Quantitative and Mathematical Sciences.

EN.600.470. Combinatorics & Graph Theory in Computer Science.
This is a graduate level course studying the applications of combinatorics and graph theory in computer science. We will start with some basic combinatorial techniques such as counting and pigeon hole principle, and then move to advanced techniques such as the probabilistic method, spectral graph theory and additive combinatorics. We shall see their applications in various areas in computer science, such as proving lower bounds in computational models, randomized algorithms, coding theory and pseudorandomness. [Analysis] Recommended Course Background: probability theory and linear algebra
Prerequisites: EN.550.171
Instructor(s): X. Li
Area: Engineering, Quantitative and Mathematical Sciences.

This is a graduate-level course studying the theoretical foundations of computer science. Topics covered will be models of computation from automata to Turing machines, computability, complexity theory, randomized algorithms, inapproximability, interactive proof systems and probabilistically checkable proofs. Students may not take both EN.600.271 and EN.600.471, unless one is for a undergrad degree and the other for grad. [Analysis]Recommended Course Background: EN.550.171 or instructor permission.
Instructor(s): X. Li
Area: Engineering, Quantitative and Mathematical Sciences.
EN.600.473. Algorithmic Game Theory.
This course provides an introduction to algorithmic game theory: the study of games from the perspective of algorithms and theoretical computer science. There will be a particular focus on games that arise naturally from economic interactions involving computer systems (such as economic interactions between large-scale networks, online advertising markets, etc.), but there will also be broad coverage of games and mechanisms of all sorts. Topics covered will include a) complexity of computing equilibria and algorithms for doing so, b) (in)efficiency of equilibria, and c) algorithmic mechanism design. [Analysis] Students may receive credit for EN.600.473 or EN.600.673, but not both.
Prerequisites: EN.600.363 OR EN.600.463 OR permission.; Students may receive credit for EN.600.473 or EN.600.673, but not both.
Instructor(s): M. Dinitz
Area: Engineering, Quantitative and Mathematical Sciences.

This course takes an application driven approach to current topics in machine learning. The course covers supervised learning (classification/structured prediction/regression/ranking), unsupervised learning (dimensionality reduction, bayesian modeling, clustering) and semi-supervised learning. Additional topics may include reinforcement learning and learning theory. The course will also consider challenges resulting from learning applications, such as transfer learning, multitask learning and large datasets. We will cover popular algorithms (naive Bayes, SVM, perceptron, HMM, winnow, LDA, k-means, maximum entropy) and will focus on how statistical learning algorithms are applied to real world applications. Students in the course will implement several learning algorithms and develop a learning system for a final project. [Applications] Recommended Course Background: multivariate calculus.
Instructor(s): I. Shpitser
Area: Engineering.

How can robots localize themselves in an environment when navigating? Can we predict which patients are at greatest-risk for complications in the hospital? Which movie should I recommend to this user given his history of likes? Many such big data questions can be answered using the paradigm of probabilistic models in machine learning. These are especially useful when common off-the-shelf algorithms such as support vector machines and k-means fail. You will learn methods for clustering, classification, structured prediction, recommendation and inference. We will use Murphy’s book, Machine Learning: a Probabilistic Perspective, as the text for this course. Assignments are solved in groups of size 1-3 students. The class will have 4 interactive sessions during which we brainstorm how to solve example open-ended real-world problems with the tools learnt in class. Students are also required to do a project of their choice within which they experiment with the ideas learnt in class. [Analysis or Applications] Students may receive credit for EN.600.476 or EN.600.676, but not both. Prerequisites include Intro Prob/Stat, Linear Algebra and Intro Machine Learning as well as strong background in s.
Instructor(s): S. Saria
Area: Engineering, Quantitative and Mathematical Sciences.

EN.600.479. Representation Learning.
Often the success of a machine learning project depends on the choice of features used. Machine learning has made great progress in training classification, regression and recognition systems when “good” representations, or features, of input data are available. However, much human effort is spent on designing good features which are usually knowledge-based and engineered by domain experts over years of trial and error. A natural question to ask then is “Can we automate the learning of useful features from raw data?” Representation learning algorithms such as principal component analysis aim at discovering better representations of inputs by learning transformations of data that disentangle factors of variation in data while retaining most of the information. The success of such data-driven approaches to feature learning depends not only on how much data we can process but also on how well the features that we learn correlate with the underlying unknown labels (semantic content in the data). This course will focus on scalable machine learning approaches for learning representations from large amounts of unlabeled, multi-modal, and heterogeneous data. We will cover topics including deep learning, multi-view learning, dimensionality reduction, similarity-based learning, and spectral learning. Students may receive credit for 600.479 or 600.679 but not both. [Analysis or Applications] Required course background: machine learning or basic probability and linear algebra.
Prerequisites: If you have completed EN.600.679 you may not enroll in EN.600.479.
Instructor(s): R. Arora
Area: Engineering.

EN.600.488. Foundations of Computational Biology & Bioinformatics II.
This course will introduce probabilistic modeling and information theory applied to biological sequence analysis, focusing on statistical models of protein families, alignment algorithms, and models of evolution. Topics will include probability theory, score matrices, hidden Markov models, maximum likelihood, expectation maximization and dynamic programming algorithms. Homework assignments will require programming in Python. Foundations of Computational Biology I is not a prereq. [Analysis] Co-listed with EN.580.488. Recommended Course Background: math through linear algebra and differential equations, at least one prob/stat course, EN.580.221 or equivalent, EN.600.226 or equivalent.
Instructor(s): R. Karchin
Area: Engineering, Natural Sciences.

EN.600.491. Computer Science Workshop I.
An applications-oriented, computer science project done under the supervision and with the sponsorship of a faculty member in the Department of Computer Science. Computer Science Workshop provides a student with an opportunity to apply theory and concepts of computer science to a significant project of mutual interest to the student and a Computer Science faculty member. Permission to enroll in CSW is granted by the faculty sponsor after his/her approval of a project proposal from the student. Interested students are advised to consult with Computer Science faculty members before preparing a Computer Science Workshop project proposal.
Instructor(s): D. Yarowsky; J. Selinski; P. Froehlich; S. Smith; Staff
Area: Engineering.
EN.600.503. Independent Study.  
Individual guided study for undergraduate students under the direction of a faculty member in the department. The program of study, including the credit to be assigned, must be worked out in advance between the student and the faculty member involved. Permission required.  
Instructor(s): Staff.

EN.600.504. Undergraduate Independent Study.  
For undergraduate students. Permission of faculty sponsor is required.  
Instructor(s): Staff.

EN.600.507. Independent Research.  
Individual research for undergraduates under the direction of a faculty member in the department. The program of research, including the credit to be assigned, must be worked out in advance between the student and the faculty member involved. Permission required.  
Instructor(s): Staff.

EN.600.508. Undergraduate Research.  
Permission of faculty sponsor is required.  
Instructor(s): Staff.

EN.600.509. Computer Science Internship.  
Individual work in the field with a learning component, supervised by a faculty member in the department. The program of study and credit assigned must be worked out in advance between the student and the faculty member involved. Students may not receive credit for work that they are paid to do. As a rule of thumb, 40 hours of work is equivalent to one credit. Permission required.  
Instructor(s): Staff.

EN.600.510. Computer Science Internship.  
Individual work in the field with a learning component, supervised by a faculty member in the department. The program of study must be worked out in advance between the student and the faculty member involved. Students may not receive credit for work that they are paid to do. As a rule of thumb, 40 hours of work is equivalent to one credit, which is the limit per semester. Permission of faculty sponsor is required.  
Instructor(s): Staff.

EN.600.519. Senior Honors Thesis.  
The student will undertake a substantial independent research project under the supervision of a faculty member, potentially leading to the notation "Departmental Honors with Thesis" on the final transcript. Students are expected to enroll in both semesters of this course during their senior year. Project proposals must be submitted and accepted in the preceding spring semester (junior year) before registration. Students will present their work publicly before April 1st of senior year. They will also submit a first draft of their project report (thesis document) at that time. Faculty will meet to decide if the thesis will be accepted for honors. Computer science majors only. Students should have a 3.5 GPA in computer science courses at the end of their junior year and permission of faculty sponsor. See EN.600.491 for faculty section numbers.  
Instructor(s): Staff.

EN.600.520. Senior Honors Thesis.  
For computer science majors only, a continuation of EN.600.519. Recommended Course Background: EN.600.519  
Instructor(s): Staff.

EN.600.546. Senior Thesis in CIS.  
Prerequisites: EN.600.445 or perm req'd.  
Instructor(s): R. Taylor  
Area: Engineering.

EN.600.550. Internship.  
Instructor(s): S. Smith; Staff.

EN.600.554. Research-Inter session.  
EN.600.556. Independent Study.  
Instructor(s): S. Smith.

Instructor(s): J. Selinski.

EN.600.578. Computer Science Workshop II.  
Permission of faculty sponsor is required.  
Instructor(s): D. Yarowsky; J. Selinski; P. Froehlich; S. Smith  
Area: Engineering.

EN.600.595. Independent Study-Summer.  
Instructor(s): Staff.

EN.600.597. Research-Summer.  
Instructor(s): Staff.

EN.600.599. Internship.  
Instructor(s): Staff.

Required for all full-time CS PhD students. Recommended for MSE students.  
Instructor(s): S. Smith.

Required for all CS PhD students. Strongly recommended for MSE students.  
Instructor(s): S. Smith.

The advent of cloud computing has lead to an explosion of storage system and data analysis software, including NoSQL databases, bulk-synchronous processing, graph computing engines, and stream processing. This course will explore scale-out software architectures for data-processing tasks. It will examine the algorithms and data-structures that underlie scalable systems and look at how hardware and networking trends influence the design and deployment of cloud computing. Recommended Course Background: EN.600.315/EN.600.415 or equivalent.  
Instructor(s): Y. Ahmad.
EN.600.625. Events Semantics in Theory and Practice.  
This course explores selected topics in the nature of event representations from the perspective of cognitive science, computer science, linguistics, and philosophy. These fields have developed a rich array of scientific theories about the representation of events, and how humans make inferences about them -- we investigate how (and if) such theories could be applied to current research topics and tasks in computational semantics such as inference from text, automated summarization, veridicality assessment, and so on. In addition to classic articles dealing with formal semantic theories, the course considers available machine-readable corpora, ontologies, and related resources that bear on event structure, such as WordNet, PropBank, FrameNet, etc.. The course is aimed to marry theory with practice: students with either a computational or linguistic background are encouraged to participate. [Applications] 
Instructor(s): B. Van Durme; K. Rawlins.

Graduate level version of EN.600.436 (see description above). Formerly EN.600.436. Students may receive credit for only one of EN.600.336, EN.600.436 or EN.600.636. Recommended Course Background: EN.600.226, AS.110.106, and Prob/Stat. 
Prerequisites: Students may receive credit for only one of EN.600.336, EN.600.436 and EN.600.636. 
Instructor(s): S. Leonard.

EN.600.638. Computational Genomics: Data Analysis.  
Graduate level version of EN.600.438. [Applications] Recommended Course Background: EN.600.226 or other programming experience, probability and statistics, linear algebra or calculus. Students may receive credit for EN.600.438 or EN.600.638 but not both. 
Prerequisites: Students may receive credit for EN.600.438 or EN.600.638, but not both. 
Instructor(s): A. Battle  
Area: Engineering.

EN.600.639. Computational Genomics.  
Graduate version of EN.600.439 [Applications] Students may earn credit for EN.600.439 or EN.600.639, but not both. Recommended Course Background: EN.600.120 and EN.600.226 
Instructor(s): B. Langmead  
Area: Engineering.

EN.600.640. Frontiers of Sequencing Data Analysis.  
Public archives now contain petabytes of valuable but hard-to-analyze DNA sequencing data. Analyzing even small datasets is complicated by sequencing errors, differences between individuals, and the fragmentary nature of the the sequencing reads. In this course, we study recent algorithms and methods that seek to make sense of DNA sequencing datasets from small to very large. Topics covered will vary from year to year, but could include RNA sequencing data analysis, other functional genomics data analysis, metagenomics analysis, data compression, indexing, applications of streaming algorithms and sketch data structures, assembly, etc. There will be homework assignments and a course project. [Applications] 
Prerequisites: EN.600.439 OR EN.600.639 
Instructor(s): B. Langmead.

Genomic data is becoming available in large quantities, but understanding how genetics contributes to human disease and other traits remains a major challenge. Machine learning approaches allow us to automatically analyze and combine genomic data, build predictive models, and identify genetic elements important to disease and cellular processes. This course will cover uses of machine learning in diverse genomic applications. Students will present and discuss current literature. Topics include predicting disease risk from genomic data, integrating diverse genomic data types, gene network reconstruction, and other topics guided by student interest. The course will include a project component with the opportunity to explore publicly available genomic data. Recommended course background: coursework in data mining, machine learning. [Applications] Students may receive credit for 600.441 or 600.641, but not both. 
Prerequisites: Students may receive credit for 600.441 or 600.641, but not both. 
Instructor(s): A. Battle.

EN.600.642. Advanced Topics in Cryptography.  
This course will focus on advanced cryptographic topics with an emphasis on open research problems and student presentations. 
Instructor(s): A. Jain.

Topics will vary from year to year, but will focus mainly on network perimeter protection, host-level protection, authentication technologies, intellectual property protection, formal analysis techniques, intrusion detection and similarly advanced subjects. Emphasis in this course is on understanding how security issues impact real systems, while maintaining an appreciation for grounding the work in fundamental science. Students will study and present various advanced research papers to the class. There will be homework assignments and a course project. 
Prerequisites: EN.600.443 OR EN.600.424 or permission of instructor. 
Instructor(s): A. Rubin.

Graduate version of EN.600.445 (see description). Students may earn credit for 600.445 or 600.645, but not both. Prerequisites: data structures and linear algebra, or permission; Recommended Course Background: intermediate programming in C/C++, 600.457, 600.461, image processing. [Applications] 
Instructor(s): R. Taylor  
Area: Engineering.

EN.600.646. Computer Integrated Surgery II.  
Students may receive credit for EN.600.446 or EN.600.646, but not both. Advanced version of EN.600.446. [Applications] 
Prerequisites: EN.600.445 OR EN.600.645 OR PERMISSION OF INSTRUCTOR 
Instructor(s): R. Taylor.

This course presents advanced methodologies and their applications to computer graphics. Recommended Course Background: any computer science courses above the 400-level in computer graphics & linear algebra; or permission of instructor.
This course will survey individual topics in computer graphics. Throughout most of the semester, students will present one to two papers in the area each week. Then, towards the end, the course will focus on the evaluation and comparison of the covered methods. Students will collaboratively design tools for analyzing the techniques and the class will jointly write up their findings in a survey paper. In addition to providing a deep study of a particular area in computer graphics, the goal of this course is to expose students to important aspects of research, including reviewing related work, designing metrics for evaluation, and writing scientific papers.
Instructor(s): M. Kazhdan.

EN.600.659. Introduction to Computational Geometry.
This course will provide an introduction to computational geometry. It will cover a number of topics in two- and three-dimensions, including polygon triangulations and partitions, convex hulls, Delaunay and Voronoi diagrams, arrangements, and spatial queries. [Analysis] Recommended Course Background: EN.600.363/463 Instructor(s): M. Kazhdan.

EN.600.660. FFT in Graphics & Vision.
In this course, we will study the Fourier Transform from the perspective of representation theory. We will begin by considering the standard transform defined by the commutative group of rotations in 2D and translations in two- and three-dimensions, and will proceed to the Fourier Transform of the non-commutative group of 3D rotations. Subjects covered will include correlation of images, shape matching, computation of invariances, and symmetry detection. Recommended Course Background: AS.110.201 and comfort with mathematical derivations.
Instructor(s): M. Kazhdan.

EN.600.661. Computer Vision.
Graduate version of EN.600.461. Students may receive credit for at most one of EN.600.361 or EN.600.461 or EN.600.661. [Applications] Prerequisites (soft): intro programming, linear algebra, and prob/stat.
Prerequisites: If you have completed EN.600.361 OR EN.600.461 you cannot enroll for EN.600.661.
Instructor(s): A. Reiter.

EN.600.662. Topics in Illumination and Reflectance Modeling for Computer Vision and Medical Imaging Applications.
The vast majority of all imagery on which computer vision is performed starts with a source of illumination in conjunction with a material reflectance property. Having a rigorous understanding of these phenomena is important for most students who want to be involved with further research in computer vision and computer integrated surgery, particularly for experimentation and development of new systems. This short course is for individuals who have already taken Computer Vision, and want to delve deeper into underlying physical modeling of light illumination, reflection, colorimetry, polarization and even sensor fusion of images taken at different Wavelengths.
Prerequisites: EN.600.361 OR EN.600.461 or Instructor permission required.
Instructor(s): L. Wolff.

EN.600.664. Randomized and Big Data Algorithms.
Graduate level version of EN.600.464. Recommended Course Background: EN.600.363 or EN.600.463. Students may receive credit for EN.600.464 or EN.600.664, but not both.
Instructor(s): V. Braverman
Area: Engineering.

EN.600.666. Information Extraction.
Introduction to statistical methods of speech recognition (automatic transcription of speech) and understanding. The course is a natural continuation of EN.600.465 but is independent of it. Topics include elementary information theory, hidden Markov models, the Baum and Viterbi algorithms, efficient hypothesis search methods, statistical decision trees, the estimation-maximization (EM) algorithm, maximum entropy estimation and estimation of discrete probabilities from sparse data for acoustic and language modeling. Weekly assignments and several programming projects. Co-listed as EN.520.666. Recommended Course Background: EN.600.120 and EN.550.310 or equivalent, expertise in C or C++ programming
Instructor(s): S. Checkoway.

The course explores the state of the art in distributed systems, networks and Internet research and practice, trying to see what it would take to push the envelop a step further. The course is conducted as a discussion group, where the professor and students brainstorm and pick interesting semester-long projects with high potential future impact. Example areas include robust scalable infrastructure (distributed datacenters, cloud networking, scada systems), real-time performance (remote surgery, trading systems), hybrid networks (mesh networks, 3G/Wifi/Bluetooth). Students should feel free to bring their own topics of interest and ideas. Recommended Course Background: a systems course (distributed systems, operating systems, computer networks, parallel programming) or permission of instructor.
Instructor(s): Y. Amir.

Topics vary but mainly focus on recent advances in exploitation techniques and defenses for software including software running on embedded systems software, browsers, and nontraditional devices such as microcontrollers in PCs. Recommended Course Background: EN.600.460 or EN.650.442 or permission of instructor
Instructor(s): S. Checkoway.

EN.600.669. Approximation Algorithms.
Graduate version of EN.600.469. Students may receive credit for EN.600.469 or EN.600.669, but not both.
Prerequisites: EN.600.363 OR EN.600.463 OR permission
Instructor(s): M. Dinitz.

EN.600.670. Pseudorandomness and Combinatorial Constructions.
Randomness is very useful in almost all areas of computer science, such as algorithms, distributed computing and cryptography. However, computers generally do not have access to truly uniform random bits. To deal with this, we rely on various pseudorandom objects to reduce either the quantity or the quality of the random bits needed. In this course, we will develop provably good pseudorandom objects for a variety of tasks. We will frequently require explicit combinatorial constructions. That is, we will want to efficiently and deterministically construct such objects. Along the way, we will also explore the close connections of such objects to many other areas in computer science and mathematics, such as graph theory, coding theory, complexity theory and arithmetic combinatorics. [Analysis] Recommended Course Background: EN.600.271/417, EN.600.363/463 and probability.
Instructor(s): X. Li.
EN.600.672. Advances in Computational Complexity.
Computational complexity theory focuses on understanding capabilities of resource bounded computation. In recent years there have been many important break-through results in this area: lower bounds for Boolean circuits, design of space efficient algorithm for undirected graph reachability problem, hardness based derandomization of randomized algorithms, probabilistic proof characterization of non-deterministic polynomial-time, to name a few. This course aims to get in detail regarding some of the very exciting recent developments in computational complexity theory. It will be useful for any student who is mathematically mature and has a keen interest in understanding the fundamental nature of computation. Recommended Course Background: EN.600.471 or equivalent.
Instructor(s): V. Vaziyam.

This is a second graduate level course in machine learning. It will provide a formal and an in-depth coverage of topics at the interface of statistical theory and computational sciences. We will revisit popular machine learning algorithms and understand their performance in terms of the size of the data (sample complexity), memory needed (space complexity), as well as the overall computational runtime (computation or iteration complexity). We will cover topics including nonparametric methods, kernel methods, online learning and reinforcement learning, as well as introduce students to current topics in large-scale machine-learning and randomized projections. Topics will vary from year-to-year but the general focus would be on combining methodology with theoretical and computational foundations. [Analysis or Applications]
Prerequisites: EN.600.475 OR EN.600.476 OR EN.600.676 OR permission of the instructor.
Instructor(s): R. Arora.

Students in the class will be asked to do assignments in Matlab. Matlab is typically easy to pick up if one is already familiar with a different programming language. Students are expected to be mathematically mature. One should have taken at least an introductory course in probability theory and linear algebra. Though not required, exposure to optimization or machine learning is recommended. Proficiency in at least one programming language is expected. When in doubt, send the instructor a copy of your transcript to see if the class is appropriate for you. Also, sit through the first few sessions and first homework to get a sense of fit. Requisites include Intro Prob/Stat, Linear Algebra and Intro Machine Learning as well as strong background in s.
Instructor(s): S. Saria.

EN.600.679. Representation Learning.
Graduate level version of 600.479. Students may receive credit for 600.479 or 600.679 but not both. [Analysis or Applications] Required course background: machine learning or basic probability and linear algebra. Co-listed with EN.600.479
Prerequisites: If you have completed EN.600.479 you may not enroll in EN.600.679.
Instructor(s): R. Arora.

EN.600.682. Deep Learning for Image Understanding.
This course discusses advanced topics on the recent progresses using deep learning, specifically deep convolutional neural networks in computer vision and medical image analysis. Topics will be selected from most recent papers from CVPR/ICCV/ArXiv/NIPS/MICCAI, with the core focus on object/scene recognition, object detection, domain transfer learning and computer-aided diagnosis. This course is targeted toward graduate students who are interested in mastering the understanding of the recent massive amount of literature and applying the skills to a course project (with lectures, paper reading, in-class presentation & discussion and a final research project).
Prerequisites: Prereq: EN.600.461 OR EN.600.661
Instructor(s): Staff.

EN.600.684. Augmented Reality.
This course introduces students to the field of Augmented Reality. It reviews its basic definitions, principles and applications. It then focuses on Medical Augmented Reality and its particular requirements. The course also discusses the main issues of calibration, tracking, multi-modal registration, advance visualization and display technologies. Homework in this course will relate to the mathematical methods used for calibration, tracking and visualization in medical augmented reality. Students may also be asked to read papers and implement various techniques within group projects. Recommended Course Background: EN.600.120, EN.600.226, and AS.110.201. [Applications]
Prerequisites: Students may take only EN.600.384 or EN.600.684, not both.
Instructor(s): N. Navab.

EN.600.688. Foundations of Computational Biology & Bioinformatics II.
This course will introduce probabilistic modeling and information theory applied to biological sequence analysis, focusing on statistical models of protein families, alignment algorithms, and models of evolution. Topics will include probability theory, score matrices, hidden Markov models, maximum likelihood, expectation maximization and dynamic programming algorithms. Homework assignments will require programming in Python. Co-listed with EN.580.688. Recommended Course Background: math through linear algebra and differential equations, at least one prob/stat course, EN.580.221 or equivalent, EN.600.226 or equivalent.
Instructor(s): R. Karchin.

In the era of data deluge, the development of methods for discovering structure in high-dimensional data is becoming increasingly important. This course will cover state-of-the-art methods from algebraic geometry, sparse and low-rank representations, and statistical learning for modeling and clustering high-dimensional data. The first part of the course will cover methods for modeling data with a single low-dimensional subspace, such as PCA, Robust PCA, Kernel PCA, and manifold learning techniques. The second part of the course will cover methods for modeling data with multiple subspaces, such as algebraic, statistical, sparse and low-rank subspace clustering techniques. The third part of the course will cover applications of these methods in image processing, computer vision, and biomedical imaging. Requisites include Linear Algebra, Optimization, and prior exposure to Machine I.
Instructor(s): R. Vidal.
**EN.600.707. Selected Topics in CS Education.**
This course will explore current issues and research in computer science education. Topics will be drawn from literature, news items, and participant experience. Current faculty and students with interests in academic careers are encouraged to attend.
Instructor(s): J. Selinski.

**EN.600.716. Selected Topics on Innovative Data Systems.**
This weekly reading group will survey and dissect the cutting-edge on innovative data systems research. Topics will encompass methods and abstraction in core systems and data management areas (e.g., cloud computing, scalable programming and storage), as well as use-cases and “war” stories from industry, and science and engineering applications. View on web: Our semester schedule is posted at damsel.cs.jhu.edu/blockparty
Instructor(s): Y. Ahnmad.

**EN.600.726. Selected Topics in Programming Languages.**
This seminar course covers recent developments in the foundations of programming language design and implementation. Topics covered include type theory, process algebra, higher-order program analysis, and constraint systems. Students will be expected to present papers orally.
Instructor(s): S. Smith.

**EN.600.728. Selected Topics in Category Theory.**
Students in this course will read a sampling of standard texts in Category Theory (e.g. the books by Awodey, Mac Lane, Pierce, or others) and papers relevant to the research of participants.
Instructor(s): N. Filardo.

**EN.600.745. Seminar in Computational Sensing and Robotics.**
This weekly seminar will focus on research issues in computer integrated surgery, including subjects such as medical image analysis, statistical modeling, visualization, vision/sensing, surgical planning, medical robotics, and clinical applications. The purpose of the course is to widen the knowledge and awareness of the participants in current research in these areas, as well as to promote greater awareness and interaction between multiple research groups within the University and beyond. The format of the course is informal presentation by a pre-eminent invited speaker, followed by free discussion. Formerly Seminar in Computer Integrated Surgery (CISST)
Instructor(s): P. Kazanzides.

**EN.600.746. Seminar: Medical Image Analysis.**
This weekly seminar will focus on research issues in medical image analysis, including image segmentation, registration, statistical modeling, and applications. It will also include selected topics relating to medical image acquisition, especially where they relate to analysis. The purpose of the course is to provide the participants with a thorough background in current research in these areas, as well as to promote greater awareness and interaction between multiple research groups within the University. The format of the course is informal. Students will read selected papers. All students will be expected to discuss these papers by the time the paper is scheduled for discussion. But individual students will be assigned on a rotating basis to lead the discussion on particular papers or sections of papers. Co-listed with En.520.746.
Instructor(s): J. Prince; R. Taylor.

**EN.600.757. Selected Topics in Computer Graphics.**
In this course we will review current research in computer graphics. We will meet for an hour once a week and one of the participants will lead the discussion for the week.
Instructor(s): M. Kazhdan.

**EN.600.760. CS Theory Seminar.**
Seminar series in theoretical computer science. Topics include algorithms, complexity theory, and related areas of TCS. Speakers will be a mix of internal and external researchers, mostly presenting recently published research papers.
Instructor(s): M. Dinitz; V. Braverman; X. Li.

**EN.600.763. Selected Topics in Streaming Algorithms.**
This course will focus on theoretical streaming algorithms and related methods. Examples of topics that we will address include sketching, sampling, frequency moments, heavy elements, k-wise independence, and the Johnson-Lindenstrauss lemma. Students will be required to select a paper and lead a discussion. This course is a good opportunity for motivated students to learn modern algorithmic methods. Recommended Course Background: EN.600.463 or equivalent.
Instructor(s): V. Braverman.

**EN.600.764. Seminar in Algorithms.**
This course will explore algorithms and theoretical computer science with a focus on algorithms for massive data. Examples of topics include streaming algorithms, approximation algorithms, online algorithms. Students will be encouraged to select a paper and lead a discussion. External speakers will be invited to present current work as well. This course is a good opportunity for motivated students to learn modern algorithmic methods. Recommended Course Background: EN.600.463 or equivalent.
Instructor(s): V. Braverman
Area: Engineering.

**EN.600.765. Selected Topics in Natural Language Processing.**
A reading group exploring important current research in the field and potentially relevant material from related fields. Enrolled students are expected to present papers and lead discussion.
Prerequisites: EN.600.465 or permission of instructor
Instructor(s): J. Eisner.

**EN.600.766. Selected Topics in Meaning, Translation and Generation of Text.**
A seminar focused on current research and survey articles on computational semantics.
Instructor(s): B. Van Durme.

**EN.600.767. Selected Topics in Systems Research.**
Students will review, present, and discuss current research in computer systems, distributed systems, and computer networks, in the contexts of dependability, performance and scalability.
Instructor(s): Y. Amir.

**EN.600.768. Selected Topics in Machine Translation.**
Students in this course will review, present, and discuss current research in machine translation. Permission of instructor.
Instructor(s): P. Koehn.

**EN.600.770. Selected Topics in Algorithms for Metric Spaces.**
This course will focus on algorithms that use, simplify, or exploit metric spaces. Examples of topics we will address include metric embeddings and their applications, algorithms to exploit low dimensionality, graph spanners and sparsifiers, and data structures such as distance oracles and compact routing schemes that allow us to efficiently find distances and paths. This course will mostly be in the form of a reading group, and students will present a paper and lead a discussion. Recommended Course Background: EN.600.363 or EN.600.463 or permission of the instructor.
Instructor(s): M. Dinitz.
Many areas of practical computer science focus on discrete data that is sequential or tree-shaped: natural language processing (sentences and their analyses), computational biology (DNA and protein structures), programming languages (computer programs and their interpretations), and compression (sequences of bits). When the data is noisy or ambiguous, decision-making requires probabilistic methods. We will survey formal tools for manipulating sets of strings, trees, sequences, and defining probabilistic models over these sets. Much of the material covers advanced topics at the intersection of formal language and automata theory, probability, and algorithms. Respectively, these three areas will enable us to represent sets, represent uncertainty, and process everything efficiently.

**EN.600.772. Selected Topics in Linear Programming and Semidefinite Programming.**
Linear programming and semi-definite programming are powerful techniques in convex optimization. They have been used to achieve the best known approximation results for many important combinatorial optimization problems, such as vertex cover, max cut, sparsest cut and MAX-2-SAT. In this course we will together explore the applications of these techniques in computer science, as well as some recent results about their limitations. Time permitting, we may also discuss their connections to the well-known unique games conjecture. This course will be in the form of a reading group, and students are encouraged to select a paper and lead a discussion. Recommended course background: EN.600.463 and EN.600.464.
Instructor(s): X. Li.

**EN.600.775. Selected Topics in Machine Learning.**
This seminar is recommended for all students interested in data intensive computing research areas (e.g., machine learning, computer vision, natural language processing, speech, computational social science). The meeting format is participatory. Papers that discuss best practices and the state-of-the-art across application areas of machine learning and data intensive computing will be read. Student volunteers lead individual meetings. Faculty and external speakers present from time-to-time. Recommended Course Background: machine learning or permission of the instructor.
Instructor(s): M. Dredze; R. Arora.

**EN.600.780. Selected Topics in Computational Genomics.**
This course will survey current areas where computer science approaches have been applied to genomics research. Chiefly, the course focuses on DNA sequencing data analysis, including sequence alignment, de novo assembly, error correction, and DNA data compression. Subject matter will be partially guided by student interests. Students will present papers orally.
Instructor(s): B. Langmead.

**EN.600.801. Dissertation Research.**
Instructor(s): Staff.

**EN.600.802. Dissertation Research.**
Instructor(s): Staff.

**EN.600.803. Graduate Research.**
Permission required. Independent research for masters or pre-dissertation PhD students.
Instructor(s): Staff.

**EN.600.804. Graduate Research.**
Independent research for masters or pre-dissertation PhD students.
Permission required.
Instructor(s): Staff.

**EN.600.807. Teaching Practicum.**
PhD students will gain valuable teaching experience, working closely with their assigned faculty supervisor. Successful completion of this course fulfills the PhD teaching requirement. (grad students) Permission req’d.
Instructor(s): J. Selinski.

**EN.600.809. Independent Study.**
Permission required. Individual study in an area of mutual interest to a graduate student and a faculty member in the department.
Instructor(s): Staff.

**EN.600.810. Graduate Independent Study.**
Permission required.
Instructor(s): Staff.

**EN.600.876. Graduate Independent Study.**
Instructor(s): D. Yarowsky; S. Smith.

**EN.600.891. Independent Study-Summer.**
Instructor(s): Staff.

**EN.600.895. Research-Summer.**
Instructor(s): Staff.

## Cross Listed Courses

### Physics Astronomy

**AS.171.205. Introduction to Practical Data Science: Beautiful Data.**
The class will provide an overview of data science, with an introduction to basic statistical principles, databases, fundamentals of algorithms and data structures, followed by practical problems in data analytics. Recommend Course Background: Familiarity with principles of computing.
Instructor(s): S. Szalay
Area: Natural Sciences, Quantitative and Mathematical Sciences.

### General Engineering

**EN.500.745. Seminar in Computational Sensing and Robotics.**
Seminar series in robotics. Topics include: Medical robotics, including computer-integrated surgical systems and image-guided intervention. Sensor based robotics, including computer vision and biomedical image analysis. Algorithmic robotics, robot control and machine learning. Autonomous robotics for monitoring, exploration and manipulation with applications in home, environmental (land, sea, space), and defense areas. Biorobotics and neuromechanics, including devices, algorithms and approaches to robotics inspired by principles in biomechanics and neuroscience. Human-machine systems, including haptic and visual feedback, human perception, cognition and decision making, and human-machine collaborative systems. Cross-listed Mechanical Engineering, Computer Science, Electrical and Computer Engineering, and Biomedical Engineering.
Instructor(s): L. Whitcomb; N. Cowan; P. Kazanzides; R. Etienne Cummings; R. Vidal.
Electrical Computer Engineering

EN.520.110. Introduction to Medical Imaging.
This course will provide the student with a basic knowledge of the principles and applications of medical imaging modalities used in biomedical patient care. Modalities include X-ray imaging, CT, ultrasound, MRI and PET/SPECT. The course will cover the underlying physics, image formation, instrumentation, and metrics used to assess the quality of medical images. The course will offer a tour and practical experience with medical imaging equipment in the clinic. Notes: This course is offered at an introductory level and assumes no prior knowledge of the material. It serves as a preparation for more complete Medical Imaging courses offered during fall and spring terms. This course is co-listed as EN.600.146.13 (Computer Science).
Instructor(s): M. Ghaly; N. Kuo
Area: Engineering.

EN.520.434. Modern Biomedical Imaging Instrumentation and Techniques.
An intermediate biomedical imaging course covering modern biomedical imaging instrumentation and techniques as applied to diagnostic radiology and other biomedical applications. It includes recent advances in various biomedical imaging modalities, multi-modality imaging and molecular imaging. The course is taught by experts in the respective fields and provides a broad based knowledge of modern biomedical imaging to prepare students for graduate studies and research in biomedical imaging. Also, the course will offer tours and practical experience with modern biomedical imaging equipments in clinical and research settings. Co-listed with EN.580.473
Prerequisites: EN.520.432 OR EN.580.472
Instructor(s): B. Tsui.

EN.520.447. Information Theory.
This course will address some basic scientific questions about systems that store or communicate information. Mathematical models will be developed for (1) the process of error-free data compression leading to the notion of entropy, (2) data (e.g. image) compression with slightly degraded reproduction leading to rate-distortion theory and (3) error-free communication of information over noisy channels leading to the notion of channel capacity. It will be shown how these quantitative measures of information have fundamental connections with statistical physics (thermodynamics), computer science (string complexity), economics (optimal portfolios), probability theory (large deviations), and statistics (Fisher information, hypothesis testing).
Instructor(s): S. Khudanpur
Area: Engineering, Quantitative and Mathematical Sciences.

EN.520.701. Current Topics in Language and Speech Processing.
This biweekly seminar will cover a broad range of current research topics in human language technology, including automatic speech recognition, natural language processing and machine translation. The Tuesday seminars will feature distinguished invited speakers, while the Friday seminars will be given by participating students. A minimum of 75% attendance and active participation will be required to earn a passing grade. Grading will be S/U.
Instructor(s): S. Khudanpur.

EN.520.702. Current Topics in Language and Speech Processing.
This biweekly seminar will cover a broad range of current research topics in human language technology, including automatic speech recognition, natural language processing and machine translation. The Tuesday seminars will feature distinguished invited speakers, while the Friday seminars will be given by participating students. A minimum of 75% attendance and active participation will be required to earn a passing grade. Cross-listed with Computer Science. Grading will be S/U.
Instructor(s): S. Khudanpur
Area: Engineering.

Mechanical Engineering

EN.530.707. Robot System Programming.
This course seeks to introduce students to open-source software tools that are available today for building complex experimental and fieldable robotic systems. The course is grouped into four sections, each of which builds on the previous in increasing complexity and specificity: tools and frameworks supporting robotics research, robotics-specific software frameworks, integrating complete robotic systems, and culminates with an independent project of the student’s own design using small mobile robots or other robots in the lab. Students will need to provide a computer or a virtual-box (with at least a few GB of memory and a few tens of GB of disc space) running Ubuntu 14.04 LTS Trusty Tahr (http://releases.ubuntu.com/14.04 or one of its variants such as Xubuntu 14.04 LTS) and ROS Indigo Igloo (http://wiki.ros.org/indigo) - note that these specific versions of Linux and ROS are required! Students should have an understanding of intermediate programming in C/C++ (including data structures and dynamic memory allocation) Familiarity with Linux programming. Familiarity with software version control systems (e.g. subversion, mercurial, git), linear algebra. Recommended Course Background: EN.530.646 Robot Devices, Kinematics, Dynamics, and Control and EN.600.636 Algorithms for Sensor Based Robotics. Students should see the course homepage http://dsci.lcsr.jhu.edu/ME530707_2016 for more information and to get started with the course. Recommended Course Background: EN.530.646 and EN.600.436.
Instructor(s): L. Whitcomb.

Biomedical Engineering

EN.580.473. Modern Biomedical Imaging Instrumentation and Techniques.
An intermediate biomedical imaging course covering modern biomedical imaging instrumentation and techniques as applied to diagnostic radiology and other biomedical applications. It includes recent advances in various biomedical imaging modalities, multi-modality imaging and molecular imaging. The course is taught by experts in the respective fields and provides a broad based knowledge of modern biomedical imaging to prepare students for graduate studies and research in biomedical imaging. Also, the course will offer tours and practical experience with modern biomedical imaging equipment in clinical and research settings. Co-listed with EN.520.434 Recommended course background: EN.520.432 or EN.580.472
Prerequisites: EN.520.432 OR EN.580.472
Instructor(s): B. Tsui
Area: Engineering, Natural Sciences.
EN.580.688. Foundations of Computational Biology & Bioinformatics II.
This course will introduce probabilistic modeling and information theory applied to biological sequence analysis, focusing on statistical models of protein families, alignment algorithms, and models of evolution. Topics will include probability theory, score matrices, hidden Markov models, maximum likelihood, expectation maximization and dynamic programming algorithms. Homework assignments will require programming in Python. Recommended Course Background: Math through linear algebra and differential equations, EN.580.221 or equivalent, EN.600.226 or equivalent.
Instructor(s): R. Karchin.

EN.580.689. Computational Personal Genomics.
What can we learn from the genome sequence of an individual? Genomic technology now makes it possible to generate huge amounts of DNA sequence data for a single individual at a relatively low cost. To make sense of this data, we need to employ sophisticated computational methods to identify genetic variations that influence an individual’s health. In this course, we will first review the state of the art in sequencing technology, and discuss how this technology is being applied to study human biology and disease. We will then explore the computational methods used to turn raw sequence data into knowledge. Topics will include genetic variant detection; discovery of chromosomal rearrangements and fusions; methods to measure gene expression from RNA; and measurements of the microbiome living inside our bodies. Recommended Course Background: EN.600.439/639, EN.600.363/463, EN.600.688, EN.580.688 (any one is sufficient), or permission of the instructor. Course is also open to undergraduate students.
Instructor(s): S. Salzberg
Area: Engineering.

EN.580.694. Statistical Connectomics.
This course will cover the basics of an exciting emerging field of statistical connectomics (aka, brain-graphs). It is so new, that we are going to make some of it up in this class! The first week will be introductory lectures that I give. The rest of the semester will be run like a seminar; each week will focus on a different topic. On Tuesdays we will hear about a statistical method that operates on graphs, and on Thursdays we will read about some neuroscience data upon which one could apply these techniques. The final project will consist of implementing a statistical method devised for graphs on a brain-graph problem. Recommended background: coursework in probability, linear algebra, and numerical programming (eg, R, Python, Matlab).
Instructor(s): J. Vogelstein
Area: Engineering.