Computer Science & Engineering

About Computer Science & Engineering

The field of Computer Science & Engineering studies the design, analysis, implementation and application of computation and computer technology. Computing plays an important role in virtually all fields, including science and medicine, music and art, business, law and human communication; hence the study of Computer Science & Engineering can be interdisciplinary in nature. Whether a student's goal is to become a computing professional or to take a few courses to develop a basic understanding of computing for application to another field, the Department of Computer Science & Engineering at Washington University is committed to helping students gain the background they need.

People are attracted to the study of computing for a variety of reasons. Consequently, the department offers a wide variety of academic programs, including a five-course minor, a second major, several undergraduate degrees, combined undergraduate and graduate programs, as well as undergraduate research opportunities and an undergraduate honors program. Each academic program can be tailored to a student's individual needs.

The breadth of Computer Science & Engineering may be best understood in terms of the general areas of software systems, hardware, theory and applications.

Software Systems are collections of interacting software components that work together to support the needs of computer applications. Courses in this area help students gain a solid understanding of how software systems are designed and implemented. Examples include operating systems, which manage computational resources; network protocols, which are responsible for the delivery of information; programming languages, which support the construction of software systems and applications; and compilers, which translate computer programs into executable form.

Hardware is the term used to describe the physical and mechanical components of a computer system. Courses in this area provide background in logic circuits, which carry out basic computations; computer architecture, which defines the organization of functional components in a computer system, and peripheral devices such as disks and robot arms that are controlled by the computer system.

Theory is the study of the fundamental capabilities and limitations of computer systems. A knowledge of theory helps students choose among competing design alternatives on the basis of their relative efficiency and helps them to verify that their implementations are correct. Theory courses provide background in algorithms, which describe how a computation is to be carried out; data structures, which specify how information is to be organized within the computer; analytical techniques to characterize the time or space requirements of an algorithm or data structure; and verification techniques to prove that solutions are correct.

Applications are the ways in which computer technology is applied to solve problems, often in other disciplines. Most applications courses provide background not only in the applications themselves but also in how the applications are designed and implemented. Examples of application areas include artificial intelligence, computer graphics and robotics.

A well-rounded study of computing includes training in each of these areas. However, depending on a student's educational goals, he or she may prefer to concentrate on certain areas for greater depth of knowledge. To help students balance their elective courses, most upper-level departmental courses are numbered with a designation in one of these categories: S for software systems, M for machines (hardware), T for theory and A for applications. If a student's interests are concentrated in the first two areas, a Computer Engineering degree might be best. If a student's interests are concentrated in the second two, a Computer Science degree might be best. Students are encouraged to meet with a faculty adviser in the Department of Computer Science & Engineering to discuss their options and develop a plan consistent with their goals.

Undergraduate Programs

These pages introduce the wide variety of undergraduate programs offered by the Department of Computer Science & Engineering and help as a starting point for students interested in studying computer science or computer engineering. We describe the minor in computer science, the second major, the pre-medical option, the BSCS degree, the BS in
Computer Engineering (see Majors (p. 5) and Minors (p. 7)), combined undergraduate and graduate programs, the Undergraduate Research Opportunities Program, and the Cooperative Education Program. Additional information can be found at http://cse.wustl.edu, and our department's faculty can offer further guidance and information about our programs.

The Majors page contains some sample schedules for several of the Computer Science & Engineering degree options. These schedules demonstrate some of the many course schedules that can be selected in completing a computer science or computer engineering degree. All students in our programs are assigned an adviser, who can help design an individualized program, monitor a student's progress, and consult about curriculum and career options.

Many nonmajors take a CSE course to broaden their education. CSE 104 Web Development, CSE 131 Computer Science I, and CSE 200 Engineering and Scientific Computing do not require any computer science background.

Dual Degree

An alternative to the second major is the dual-degree program that leads to two undergraduate degrees, one in the School of Engineering & Applied Science and one from another school. For this option a student must complete all requirements (including distribution requirements) for both degrees. If Arts & Sciences is the other school, 150 credits are required for two degrees. (Only 120 credits are required for the second major because a single degree is earned.) For details, see the Engineering Student Services Office: Lopata Hall, Room 303, (314) 935-6100.

Pre-medical Option within Computer Science

Students may pursue a pre-medicine curriculum in conjunction with either the BS degree or second major in computer science programs. Students interested in the pre-medical option should refer to the School of Engineering Bulletin page for details.

Combined Undergraduate and Graduate Study

The Department of Computer Science & Engineering offers in-depth graduate study in many areas. Students entering the graduate programs require a background in computer science fundamentals equivalent to at least the minor in computer science. Washington University undergraduates seeking admission to the graduate degree program to obtain a master's degree in computer science or computer engineering do not need to take the Graduate Record Examination (GRE). For more information, contact the department office at (314) 935-6160 or the associate chair at associatechair@cse.wustl.edu.

The Joint BS–MS

This five-year program that leads to both the bachelor's and master's degrees offers the student an excellent opportunity to combine undergraduate and graduate studies in an integrated curriculum. The combination of the two programs extends the flexibility of the undergraduate curriculum to more advanced studies, thereby enabling students to plan their entire spectrum of computing studies in a more comprehensive educational framework. Consistent with the general requirements defined by the School of Engineering, a minimum of 150 units is required for completion of the BS–MS program. Provided that the 150-unit requirement is satisfied, up to 6 units of course work acceptable for the MS can be counted toward both the BS and MS requirement. Students in the BS–MS program can take advantage of the program's flexibility by taking graduate courses toward the graduate degree while still completing the undergraduate degree requirements.

The BS–MS program offers early admission to the graduate programs in computer science and computer engineering in the junior year and allows a student to complete the master's degree typically in only one additional year of study (instead of the usual three semesters). Undergraduate financial support is not extended for the additional semesters to complete the master's degree requirements; however, scholarship support based upon a student's major GPA, calculated at the end of the junior year, will be awarded automatically in the student's final year of study. Students are classified as graduate students in their final year of study, and their tuition charges are at the graduate student rate.

For information regarding scholarship amounts, please visit the BS–MS Program in Engineering web page.

If students plan to apply to this program, it is recommended that they complete at least an undergraduate minor in computer science, three additional computer science courses at the 400 level, and one at the 500 level during their first four years. Students must apply to this program by September 1 of their senior year, and a minimum GPA of 3.0 is required of all applicants.

BS–MBA Program

The growing importance of computer-based information systems in the business environment has produced a sustained high demand for graduates with master's degrees in business administration and undergraduate majors in computer science and engineering.

Students should apply to this joint program by February 1 of their junior year. The application for admission to the Olin Business School is available through the business school. Applicants are judged on undergraduate performance, GMAT scores, summer and/or co-op work experience, recommendations and a personal interview.
Research and Industry Experience

If a student wants to become involved in computer science or computer engineering research or gain experience in industry while he or she is an undergraduate, there are many opportunities to do so. A few of these are listed below.

Co-op: The Cooperative Education Program allows a student to get valuable experience working in industry as an undergraduate. Doing a Co-op can help give students another perspective on their education and may lead to full-time employment. Numerous companies participate in this program. More information is available from the Engineering Co-op and Internship Program that is part of the Career Center in the Danforth University Center, Suite 110.

Research: Participating in an undergraduate research project is a great way to learn a lot about a specific area and find out what research is all about. Research projects are available either for pay or for credit through CSE 400E Independent Study. Visit our website (http://cse.wustl.edu/undergraduateprograms/Pages/undergraduate-research-opportunities.aspx) for details.

Honors Program: If a student maintains a 3.5 GPA and completes a thesis describing research that he or she has performed while an undergraduate, then the student will graduate "with distinction" and the thesis title will appear on the student's transcript. Besides being a valuable experience, completing a thesis can be advantageous if the student applies to graduate school. When selecting this option, the student should register for CSE 499 Undergraduate Honors Thesis. Students in the Honors Program are encouraged to take several graduate-level courses and to serve as undergraduate teaching assistants for upper-level undergraduate courses.

Computing Facilities

The School of Engineering & Applied Science has numerous undergraduate laboratories that are accessible to all students regardless of their majors. In addition, the Department of Computer Science & Engineering maintains specialized instructional laboratories such as those in support of computer engineering classes. The department actively promotes a culture of strong undergraduate participation in research. Many undergraduates work in research labs with state-of-the-art equipment that provides them the opportunity to take part in computer science and computer engineering research. Sensor networks, high-speed routers, specialized FPGA hardware, wireless devices, RF tags, digital cameras, robots, large displays and multiprocessors are just a few of the hardware devices undergraduates often use in their projects. Opportunities for exploring modern software development techniques and specialized software systems further enrich the range of research options and help undergraduates sharpen their design and programming skills.

Advanced Placement/Proficiency

Students receiving a 4 or 5 on the AP Computer Science A exam are awarded credit for CSE 131 Computer Science I if they pass the CSE 131 proficiency exam. Any student can take the CSE 131 proficiency exam, and a suitable score will waive CSE 131 as a requirement.

Upon request, the computer science department will evaluate a student for proficiency for any of our introductory courses. If a student is determined to be proficient in a given course, that course will be waived (without awarding credit) in the student's degree requirements, and the student will be offered guidance in selecting a more advanced course. Questions should be directed to the associate chair at associatechair@cse.wustl.edu.

Undergraduate Courses

Course requirements for the minor and majors may be fulfilled by CSE 131 Computer Science I, CSE 132 Computer Science II, CSE 240 Logic and Discrete Mathematics, CSE 241 Algorithms and Data Structures, and CSE courses with a letter suffix in any of the categories: software systems (S), hardware (M), theory (T) and applications (A). In addition, with approval of the instructor, up to 6 units of CSE 400E Independent Study can be used toward the CSE electives of any CSE degree. Other CSE courses provide credit toward graduation but not toward the CSE elective requirements for the second major, BSCS or BSCO. Undergraduates are encouraged to consider 500-level courses. If a student is interested in taking a course but is not sure if he or she has the needed prerequisites, the student should contact the instructor.

Phone: (314) 935-6160
Departmental website: http://cse.wustl.edu/undergraduateprograms

Endowed Professors

Aaron Bobick
James M. McKelvey Professor and Dean
PhD, Massachusetts Institute of Technology
Computer vision, graphics, human-robot collaboration

Michael R. Brent
Henry Edwin Sever Professor of Engineering
PhD, Massachusetts Institute of Technology
Systems biology, computational and experimental genomics, mathematical modeling, algorithms for computational biology, bioinformatics
Chenyang Lu  
Fullgraf Professor in the Department of Computer Science & Engineering  
PhD, University of Virginia  
Real-time and embedded systems, wireless sensor networks, mobile computing

**Professors**

**Jeremy Buhler**  
PhD, Washington University  
Computational biology, genomics, algorithms for comparing and annotating large biosequences

**Shantanu Chakrabarthy**  
PhD, Johns Hopkins University  
Analog computing techniques, self-powered sensors, floating-gate circuits, biosensors and bioelectronics

**Roger D. Chamberlain**  
DSc, Washington University  
Computer engineering, parallel computation, computer architecture, multiprocessor systems

**Ron K. Cytron**  
PhD, University of Illinois at Urbana-Champaign  
Programming languages, middleware, real-time systems

**Christopher D. Gill**  
DSc, Washington University  
Distributed real-time embedded systems, middleware, formal models and analysis of concurrency and timing

**Raj Jain**  
PhD, Harvard University  
Wireless networks, network security, next generation Internet, sensor networks, telecommunications networks, performance analysis, traffic management, quality of service

**Robert Pless**  
PhD, University of Maryland  
Computer vision, medical imaging, sensor network algorithms, citizen science

**Weixiong Zhang**  
PhD, University of California, Los Angeles  
Computational biology, genomics, machine learning and data mining, and combinatorial optimization

**Associate Endowed Professor**

**Caitlin Kelleher**  
Hugo F. & Ina Champ Urbauer Career Development Associate Professor  
PhD, Carnegie Mellon University  
Human-computer interaction, programming environments, and learning environments

**Associate Professors**

**Kunal Agrawal**  
PhD, Massachusetts Institute of Technology  
Parallel computing, cyber-physical systems & sensing, theoretical computer science

**Yixin Chen**  
PhD, University of Illinois at Urbana-Champaign  
Mathematical optimization, artificial intelligence, planning and scheduling, data mining, learning data warehousing, operations research, data security

**Patrick Crowley**  
PhD, University of Washington  
Computer and network systems, network security

**Sanmay Das**  
PhD, Massachusetts Institute of Technology  
Design of algorithms for complex environments, computational social science, machine learning

**Viktor Gruev**  
PhD, Johns Hopkins University  
Low power integrated sensory systems, integrated polarization imaging, focal plane spatiotemporal image sensors, current mode image sensors, sensory systems in 3-D fabrication technology, micro/nano fabrication, micro fluids, and low power analog/digital integrated circuits

**Tao Ju**  
PhD, Rice University  
Computer graphics, visualization, mesh processing, medical imaging and modeling

**William D. Richard**  
PhD, University of Missouri–Rolla  
Ultrasonic imaging, medical instrumentation, computer engineering

**Assistant Professors**

**Yasutaka Furukawa**  
PhD, University of Illinois at Urbana-Champaign  
Computer vision and computer graphics

**Roman Garnett**  
PhD, University of Oxford  
Active learning (especially with atypical objectives), Bayesian optimization, and Bayesian nonparametric analysis

**Brendan Juba**  
PhD, Massachusetts Institute of Technology  
Theoretical approaches to artificial intelligence founded on computational complexity theory and theoretical computer science more broadly construed
Angelina Lee  
PhD, Massachusetts Institute of Technology  
Designing linguistics for parallel programming, developing runtime system support for multithreaded software, and building novel mechanisms in operating systems and hardware to efficiently support parallel abstractions

Benjamin Moseley  
PhD, University of Illinois at Urbana-Champaign  
Design and analysis of algorithms, online and approximation algorithms, parallel computing, large data analysis, green computing and algorithmic applications

Research Faculty

Sharlee Climer  
PhD, Washington University  
Computational biology, artificial intelligence, mathematical modeling, combinatorial optimization, pattern recognition

Lecturers

Ruth Miller  
PhD, University of Houston  
Data mining, database, bioinformatics

Marion Neumann  
PhD, University of Bonn  
Machine learning with graphs; solving problems in agriculture and robotics

Douglas Shook  
MS, Washington University  
Imaging sensor design, compiler design and optimization

Todd Sproull  
PhD, Washington University  
Computer networking and mobile application development

Senior Professors

Jerome R. Cox Jr.  
ScD, Massachusetts Institute of Technology  
Computer system design, computer networking, biomedical computing

Mark A. Franklin  
Hugo F. and Ina Champ Urbauer Professor of Engineering  
PhD, Carnegie Mellon University  
Computer architecture, systems analysis and parallel processing, storage systems design

Jonathan S. Turner  
PhD, Northwestern University  
Design and analysis of internet routers and switching systems, networking and communications, algorithms

Professors Emeriti

Richard A. Dammkoehler  
MS, Washington University  
Computer programming theory, information retrieval, computer systems architecture

Takayuki D. Kimura  
PhD, University of Pennsylvania  
Communication and computation, visual programming

Seymour V. Pollack  
MS, Brooklyn Polytechnic Institute  
Intellectual property, information systems

Majors

Please see the sections below for information about the Bachelor of Science in Computer Science (p. 5), the Bachelor of Science in Computer Engineering (p. 6), and the second major in computer science (p. 7).

Bachelor of Science in Computer Science

The Bachelor of Science in Computer Science (BSCS) is designed for students planning a career in computing. Students working toward a BSCS degree must meet all requirements for an applied science degree from the School of Engineering & Applied Science. In addition, there are the following departmental course requirements:

- **Computer Science Core Requirements**
  - CSE 131  
    - Computer Science I  
    - 3 units
  - CSE 132  
    - Computer Science II  
    - 3 units
  - CSE 240  
    - Logic and Discrete Mathematics  
    - 3 units
  - or Math 310  
    - Foundations for Higher Mathematics  
    - 3 units
  - CSE 241  
    - Algorithms and Data Structures  
    - 3 units
  - CSE 332S  
    - Object-Oriented Software Development Laboratory  
    - 3 units

**Total units**: 15

Each of these core courses must be passed with a grade of C– or better.

- **Computer Science Technical Elective Requirements**:  
  At least 27 additional units in computer science or computer science-related courses with an S, M, T or A suffix of which at least one must be a theory (T) course, at least one must be a systems (S) course, and at least one must be a machine (M) or application (A) course. Students may use up to 6 units of approved independent work (CSE 400E, CSE 497-CSE 499) as part of their computer science electives. Such independent work is not classified as S, M, T or A.
There are several technical elective course sequences described on the department webpage that are recommended for students whose interests are in particular areas of computing.

- **Math Requirements:**
  - Math 131 Calculus I 3
  - Math 132 Calculus II 3
  - Math 233 Calculus III 3
  - Math 309 Matrix Algebra 3
  - or ESE 318 Engineering Mathematics A 3
  - ESE 326 Probability and Statistics for Engineering 3
  - or Math 3200 Elementary to Intermediate Statistics and Data Analysis 3

  **Total units:** 15

Upon completing a course in the calculus sequence (Math 131–Math 132–Math 233) with a grade of C– or better, the student may apply to receive credit for the preceding courses in the calculus sequence.

- **Additional Departmental Requirements:**
  - CWP 100 College Writing 1 3
  - Engr 310 Technical Writing 3
  - Natural Sciences electives 8
  - Humanities and Social Sciences electives 18

The College Writing and Humanities and Social Sciences requirements are those required of all students in the School of Engineering & Applied Science. The Natural Sciences requirement is for 8 units designated NSM (Natural Sciences and Mathematics) from any of the following departments: Anthropology, Biology, Chemistry, Earth and Planetary Sciences, or Physics. The College Writing and Natural Sciences courses must be completed with a grade of C– or better.

All courses taken to meet any of the above requirements (with the exception of the humanities and social sciences electives) cannot be taken on a pass/fail basis.

See a sample schedule for Computer Science Degree Options in this Bulletin.

**Bachelor of Science in Computer Engineering**

Computer Engineering encompasses studies of hardware, software and systems issues that arise in the design, development and application of computer systems. Computer Engineering students choose between a more hardware-focused or a more software-focused degree program, each with slightly different requirements.

Students working toward a BScOE degree must meet all requirements for an engineering degree from the School of Engineering & Applied Science. Required courses, focus area courses, technical electives, and the senior project cannot be taken on a pass/fail basis. In addition, there are the following departmental course requirements:

- **Common Studies Program Requirements:**
  - Math 131 Calculus I 3
  - Math 132 Calculus II 3
  - Math 217 Differential Equations 3
  - Math 233 Calculus III 3
  - Physics 117A General Physics I 4
  - or Physics 197 Physics I 4
  - Physics 118A General Physics II 4
  - or Physics 198 Physics II 4
  - Natural Sciences elective 3
  - CWP 100 College Writing 1 (unless waived) 3
  - Engr 310 Technical Writing 3
  - Humanities and social sciences electives 18

  **Total units:** 47

Upon completing a course in the calculus sequence (Math 131–Math 132–Math 233) with a grade of C– or better, the student may apply to receive credit for the preceding courses in the calculus sequence.

The Natural Sciences requirement is for 3 units designated NSM (Natural Sciences and Mathematics) from any of the following departments: Anthropology, Biology, Chemistry, Earth and Planetary Sciences, or Physics. This replaces the Chemistry requirement in the Engineering Common Studies list. These courses must be completed with a grade of C– or better.

The College Writing and Humanities and Social Sciences requirements are those required of all students in the School of Engineering & Applied Science.

- **Computer Engineering Core Requirements:**
  - CSE 131 Computer Science I 3
  - CSE 132 Computer Science II 3
  - ESE 230 Introduction to Electrical and Electronic Circuits 4
  - ESE 232 Introduction to Electronic Circuits 3
  - CSE 240 Logic and Discrete Mathematics 3
  - CSE 241 Algorithms and Data Structures 3
  - CSE 260M Introduction to Digital Logic and Computer Design 3
  - ESE 326 Probability and Statistics for Engineering 3
  - CSE 361S Introduction to Systems Software 3
  - CSE 362M Computer Architecture 3

  **Total units:** 31

Each of these core courses must be passed with a grade of C– or better.
• Computer Engineering Focus Area:
  Each student must satisfy the requirements of at least one focus area.
  **Hardware Focus:**
  CSE 462M  Computer Systems Design  3
  and one of the following
  CSE 463M  Digital Integrated Circuit Design and Architecture  3
  CSE 464M  Digital Systems Engineering  3
  CSE 465M  Digital Systems Laboratory  3
  **Software Focus:**
  CSE 422S  Operating Systems Organization  3
  and one of the following
  CSE 431S  Translation of Computer Languages  3
  CSE 467S  Embedded Computing Systems  3
  CSE 473S  Introduction to Computer Networks  3

• Computer Engineering Technical Electives:
  At least 21 units of technical electives, drawn from 300-level or higher CSE (with suffix S, M, T, or A) or ESE courses. These units are in addition to the 6 units needed to meet the focus area requirement described above.

• Computer Engineering Senior Project:
  Each student must complete a senior project (CSE 497, 3 credits), either individually or as part of a group under the direction of a CSE or ESE professor.

See a sample schedule for Computer Engineering Degree Options in this *Bulletin*.

**The Second Major in Computer Science**

The second major provides an opportunity to combine computer science with another degree program. A second major in computer science can expand a student's career options and enable interdisciplinary study in areas such as cognitive science, computational biology, chemistry, physics, philosophy and linguistics. The second major also is well suited for students planning careers in medicine, law, business, architecture and fine arts. The requirements are as follows. There are no additional distribution or unit requirements for the second major.

• **Computer Science Core Requirements:**
  CSE 131  Computer Science I  3
  CSE 132  Computer Science II  3
  CSE 240  Logic and Discrete Mathematics  3
  or Math 310  Foundations for Higher Mathematics
  CSE 241  Algorithms and Data Structures  3

CSE 322S  Object-Oriented Software Development Laboratory  3

**Total units**  15

Each of these core courses must be passed with a grade of C− or better.

• **Computer Science Electives:**
  At least 15 units of computer science electives, selected from any CSE courses with an S, M, T or A suffix.

• **Math Requirement:**
  Calculus (Math 131) and Probability (ESE 326 or Math 3200, or the sequence QBA 120-QBA 121).

• **Capstone:**
  An additional 6 units of course work (or independent study) at the 300 level or higher with a significant computational component. The capstone may be completed in any department and provides an opportunity for interdisciplinary study, such as a thesis that applies computer science to another field. A student’s CSE adviser must approve the capstone in advance.

**Minors**

Please see the sections below for information about the minor in computer science (p. 7) and the minor in bioinformatics (p. 8).

**The Minor in Computer Science**

If a student’s goal is a basic foundation in computer science for application to another field, but he or she is not planning a career as a practicing computer scientist, the minor in computer science is a good choice. The minor consists of five CSE courses, including three core courses and two electives. The core courses provide an introduction to computer science concepts and problem-solving techniques. The electives offer flexibility to integrate computer science studies with a major area. Students select the courses that are most important to them, whether their interests are in fine arts, linguistics, psychology, philosophy, law, business, medicine, the natural sciences or anything else.

**Units required:** 15

**Required courses:**

<table>
<thead>
<tr>
<th>Course</th>
<th>Title</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSE 131</td>
<td>Computer Science I</td>
<td>3</td>
</tr>
<tr>
<td>CSE 132</td>
<td>Computer Science II</td>
<td>3</td>
</tr>
<tr>
<td>CSE 241</td>
<td>Algorithms and Data Structures</td>
<td>3</td>
</tr>
</tbody>
</table>

**Total units**  9

**Elective courses:**

Any two additional CSE courses selected among CSE 200 Engineering and Scientific Computing, CSE 240 Logic and
Discrete Mathematics, and any CSE course with a T, S, M or A suffix.

**Additional Information**

All courses used for the computer science minor must be taken for a grade, and the student must earn a C– or better. Should the student decide to go further in the field, all courses in the computer science minor, except CSE 200, can be used toward a second major in computer science or a degree in computer science or computer engineering.

**The Minor in Bioinformatics**

Mindful of the emerging opportunities at the interface of biology and computer science, the Department of Biology and Department of Computer Science & Engineering have fashioned the minor in bioinformatics that serves students from both departments as well as other students from the natural sciences and engineering with an interest in this field.

**Units required:** 23–24 units as described below

**Core courses:**

<table>
<thead>
<tr>
<th>Course</th>
<th>Title</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biol 2960</td>
<td>Principles of Biology I</td>
<td>4</td>
</tr>
<tr>
<td>Biol 2970</td>
<td>Principles of Biology II</td>
<td>4</td>
</tr>
<tr>
<td>CSE 241</td>
<td>Computer Science I</td>
<td>3</td>
</tr>
<tr>
<td>Math 2200</td>
<td>Elementary Probability and Statistics</td>
<td>3</td>
</tr>
<tr>
<td>or Math 3200</td>
<td>Elementary to Intermediate Statistics and Data Analysis</td>
<td>3</td>
</tr>
<tr>
<td>or ESE 326</td>
<td>Probability and Statistics for Engineering</td>
<td></td>
</tr>
</tbody>
</table>

Total units: 17

**Advanced biology elective:**

Choose one of:

<table>
<thead>
<tr>
<th>Course</th>
<th>Title</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biol 3492</td>
<td>Laboratory Experiments with Eukaryotic Microbes</td>
<td>3</td>
</tr>
<tr>
<td>Biol 4181</td>
<td>Population Genetics</td>
<td>3</td>
</tr>
<tr>
<td>Biol 4342</td>
<td>Research Explorations in Genomics</td>
<td>4</td>
</tr>
<tr>
<td>Biol 437</td>
<td>Laboratory on DNA Manipulation</td>
<td>4</td>
</tr>
</tbody>
</table>

**CSE elective:**

Choose one of:

<table>
<thead>
<tr>
<th>Course</th>
<th>Title</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSE 514A</td>
<td>Data Mining</td>
<td>3</td>
</tr>
<tr>
<td>CSE 584A</td>
<td>Algorithms for Biosequence Comparison</td>
<td>3</td>
</tr>
<tr>
<td>CSE 587A</td>
<td>Algorithms for Computational Biology</td>
<td>3</td>
</tr>
</tbody>
</table>

**Additional Information**

It is anticipated that for those students majoring in biology or CSE, some portion of the introductory sequence will overlap with courses required for the major, and these courses will be applicable to both the major and the minor. Upper-level courses in Biology and CSE used to fulfill the minor may not be used to fulfill another major or minor in Arts & Sciences. A minimum grade of C– is required for all courses to count toward the minor.

**Courses**

Visit https://courses.wustl.edu to view semester offerings for E81 CSE.

---

**E81 CSE 104 Web Development**

This comprehensive course does not assume prior programming background or web design experience. Explores elementary concepts, graphic design, HTML, CSS, basic scripting techniques, search engine optimization and site maintenance/marketing strategies. Credit 3 units. EN: TU

---

**E81 CSE 131 Computer Science I**

An introduction to software concepts and implementation, emphasizing problem solving through abstraction and decomposition. Introduces processes and algorithms, procedural abstraction, data abstraction, encapsulation and object-oriented programming. Recursion, iteration and simple data structures are covered. Concepts and skills are mastered through programming projects, many of which employ graphics to enhance conceptual understanding. Java, an object-oriented programming language, is the vehicle of exploration. Active-learning sessions are conducted in a studio setting in which students interact with each other and the professor to solve problems collaboratively. Prerequisites: Comfort with algebra and geometry at the high school level is assumed. Patience, good planning and organization promote success. This course assumes no prior experience with programming. Credit 3 units. BU: SCI EN: TU

---

**E81 CSE 131R Seminar: Computer Science I**

A seminar and discussion session that complements the material studied in CSE 131. Provides background and breadth for the disciplines of computer science and computer engineering. Features guest lectures and highly interactive discussions of diverse computer science topics. Highly recommended for majors and for any student seeking a broader view of computer science or computer engineering. Pass/Fail only. Credit 1 unit.

---

**E81 CSE 132 Computer Science II**

CSE 132 introduces students to fundamental concepts in the basic operation of computers, ranging from desktops and servers to microcontrollers and handheld devices. Subjects include digital and analog input/output, sensing the physical world, information representation, basic computer architecture and machine language, time-critical computation, machine-to-machine communication and protocol design. Students use both desktop systems and handheld microcontrollers for laboratory experiments. Active-learning sessions are conducted in a studio setting in which students interact with each other and the professor to solve problems collaboratively. Prerequisite: CSE 131 or equivalent. Credit 3 units. EN: TU
E81 CSE 200 Engineering and Scientific Computing
This course provides an introduction to numerical methods for scientific computation that are relevant to engineering problems. Topics addressed include interpolation, integration, linear systems, least-squares fitting, nonlinear equations and optimization and initial value problems. Basic procedural programming concepts (procedural and data abstraction, iteration, recursion) are covered using MATLAB. C is briefly covered so the students understand that the algorithms and programming concepts apply in both. Prerequisite: Math 132. Credit 3 units. EN: TU

E81 CSE 220S Software Design and Development Studio
This is a lab course that provides practical experience in designing, implementing, testing, documenting and supporting a medium-sized software application. Topics covered include application and user interface specification, module and API design, code re-use, code review, software maintenance and support, unit and integration testing, and debugging procedures. Students gain experience in the application of common algorithms, design patterns and data structures to novel problems. Students have a choice of working in Java or C++, and work both individually and in groups. Specific application areas vary by semester. Prerequisites: sophomore standing and CSE 132. CSE 241 is recommended. Credit 3 units. EN: TU

E81 CSE 232 Programming Skills Workshop
This course provides an overview of practical implementation skills. Topics include compilation and linking, memory management, pointers and references, using code libraries, testing and debugging. Prerequisites: CSE 132. Credit 1 unit.

E81 CSE 240 Logic and Discrete Mathematics
Introduces elements of logic and discrete mathematics that allow reasoning about computational structures and processes. Generally, the areas of discrete structures, proof techniques and computational models are covered. Topics typically include propositional and predicate logic; sets, relations, functions and graphs; proof by contradiction, induction and reduction; and finite state machines and regular languages. Prerequisite: CSE 131 or other introductory programming background. Credit 3 units. EN: TU

E81 CSE 241 Algorithms and Data Structures
Study of fundamental algorithms, data structures and their effective use in a variety of applications. Emphasizes importance of data structure choice and implementation for obtaining the most efficient algorithm for solving a given problem. A key component of this course is worst-case asymptotic analysis, which provides a quick and simple method for determining the scalability and effectiveness of an algorithm. Other topics covered generally include: divide-and-conquer algorithms, sorting algorithms, decision tree lower-bound technique, hashing, binary heaps, skip lists, B-trees, basic graph algorithms. Prerequisites: CSE 131, CSE 240 (or some basic discrete mathematics background) is strongly recommended. Credit 3 units. EN: TU

E81 CSE 245A Fair Division in Theory and Practice
The concept of fair division is a central tenet in the design of procedures aimed at generating equitable social outcomes and mitigating conflict. At the national level, such procedures include systems of apportionment, voting and legislative districting, to name a few. On a smaller scale, these procedures could govern how assets are divided in a divorce, or how to divide a cake. While "fairness" in theory is indisputably a good thing, in practice the courts, politicians, and even mathematicians have grappled with the question of what it means for a procedure to be fair. This course examines algorithms and applications of procedures that aim to divide or allocate resources fairly. Some of these procedures were developed by mathematicians looking for formulas that satisfy mathematical properties such as envy-freeness and equitability. All of the procedures we consider are examined in terms of the fairness goals they aspire to achieve, the mechanisms they employ to achieve those goals, and the shortcomings of the procedures. Same as InterD 245A Credit 3 units. A&S: SS & A&S IQ: SSC EN: TU

E81 CSE 260M Introduction to Digital Logic and Computer Design
Introduction to design methods for digital logic and fundamentals of computer architecture. Boolean algebra and logic minimization techniques; sources of delay in combinational circuits and effect on circuit performance; survey of common combinational circuit components; sequential circuit design and analysis; timing analysis of sequential circuits; use of computer-aided design tools for digital logic design (schematic capture, hardware description languages, simulation); design of simple processors and memory subsystems; program execution in simple processors; basic techniques for enhancing processor performance; configurable logic devices. Prerequisites: CSE 131 or comparable programming experience. Credit 3 units. EN: TU

E81 CSE 316A Social Network Analysis
This course looks at social networks and markets through the eyes of a computer scientist. We look at questions including, "Why are acquaintances rather than friends more likely to get us job opportunities?" and, "Why do the rich get richer?" We begin by studying graph theory (allowing us to study the structure) and game theory (allowing us to study the interactions) of social networks and market behavior at the introductory level. Among other topics, we study auctions, epidemics, and the structure of the Internet (including web searches). This course examines the intersection of computer science, economics, sociology, and applied mathematics. Prerequisites: CSE 241 and basic knowledge of probability and statistics. Credit 3 units. EN: TU

E81 CSE 320S Software Design and Development Studio
This is a lab course that provides practical experience in designing, implementing, testing, documenting and supporting a medium-sized software application. Topics covered include application and user interface specification, module and API design, code re-use, code review, software maintenance and support, unit and integration testing, and debugging procedures. Students gain experience in the application of common algorithms, design patterns and data structures to novel problems. Students have a choice of working in Java or C++, and work both individually and in groups. Specific application
areas vary by semester. Prerequisites: junior standing and CSE 132. CSE 241 is recommended.
Credit 3 units. EN: TU

E81 CSE 330S Rapid Prototype Development and Creative Programming
This course uses web development as a vehicle for developing skills in rapid prototyping. Students acquire the skills to build a Linux web server in Apache, to write a web site from scratch in PHP, to run an SQL database, to perform scripting in Python, to employ the Django web framework and to develop modern web applications in client-side and server-side JavaScript. The course culminates with a creative project in which students are able to synthesize the course material into a project of their own interest. The course implements an interactive studio format: after a formal presentation of a topic, students develop a related project under the supervision of the instructor. Prerequisite: CSE 131 or an equivalent programming background.
Credit 3 units. EN: TU

E81 CSE 332S Object-Oriented Software Development Laboratory
Intensive focus on practical aspects of designing, implementing and debugging software, using object-oriented, procedural, and generic programming techniques. The course emphasizes familiarity and proficiency with a wide range of C++ language features through hands-on practice completing studio exercises and lab assignments, supplemented with readings and summary presentations for each session. Prerequisites: CSE 132 or CSE 241.
Credit 3 units. EN: TU

E81 CSE 341T Parallel and Sequential Algorithms
The course aims to teach how to design, analyze and implement parallel and sequential algorithms. The emphasis is on teaching fundamental principles and design techniques that easily transfer over to parallel programming. These techniques include divide and conquer, contraction, the greedy method, etc. The course does have a significant programming component using Cilk Plus, a parallel programming language. Prerequisite: CSE 240. CSE 241 is recommended, but the course can be taken without it with instructor permission.
Credit 3 units. EN: TU

E81 CSE 361S Introduction to Systems Software
Introduction to the hardware and software foundations of computer processing systems. Examines the process whereby computer systems manage, interpret and execute applications. Covers fundamental algorithms for numerical computation, memory organization and access, storage allocation, and the sequencing and control of peripheral devices. Weekly laboratories, exercises and a final laboratory project.
Prerequisite: CSE 131.
Credit 3 units. EN: TU

E81 CSE 362M Computer Architecture
Study of interaction and design philosophy of hardware and software for digital computer systems. Processor architecture, Instruction Set Architecture, Assembly Language, memory hierarchy design, I/O considerations. Comparison of computer architectures. Prerequisite: CSE 260M.
Credit 3 units. EN: TU
E81 CSE 425S Programming Systems and Languages
A systematic study of the principles, concepts and mechanisms of computer programming languages: their syntax, semantics and pragmatics; the processing and interpretation of computer programs; programming paradigms; and language design. Illustrative examples are selected from a variety of programming language paradigms. Prerequisites: CSE 132, CSE 240 and CSE 241.
Credit 3 units. EN: TU

E81 CSE 426S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 427S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 428S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 429S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 430S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 431S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 432S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 433S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 434S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 435S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 436S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 437S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 438S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 439S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 440S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 441S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 442S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 443S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 444S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 445S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 446S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 447S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 448S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 449S Programming Systems and Languages
A study of the techniques and methodologies used in the design and implementation of software systems. Topics include: software design methods, software engineering process models, software project management, software quality assurance, and software evolution. Prerequisites: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 450A Video Game Programming
This course teaches the core aspects of a video game developer's toolkit. Students work in groups and with a large game software engine to make a full-featured video game. Students have the opportunity to work on topics in graphics, artificial intelligence, networking, physics, user interface design and other topics. Prerequisite: CSE 332S.
Credit 3 units. EN: TU

E81 CSE 452A Computer Graphics
Introduction to computer graphics. Input, representation, manipulation and display of geometric information. Two-dimensional display of three-dimensional objects: perspective, hidden surface, shading, animation. Display and input devices. Issues in designing interactive graphics systems. Issues in building 3-dimensional renderers. Students develop interactive graphics programs with a standard graphics package and using various graphics input and output devices. Prerequisites: CSE 332S and basic knowledge in linear algebra.
Credit 3 units. EN: TU

E81 CSE 460T Switching Theory
Advanced topics in switching theory as employed in the synthesis, analysis and design of information processing systems. Combinational techniques: minimization, multiple output networks, state identification and fault detection, hazards, testability and design for test are examined. Sequential techniques: synchronous circuits, machine minimization, optimal state assignment, asynchronous circuits and built-in self-test techniques. Prerequisite: CSE 250M or equivalent.
Credit 3 units. EN: TU

E81 CSE 462M Computer Systems Design
Introduction to modern design practices, including the use of FPGA design methodologies. Students use a commercial CAE/CAD system for VHDL-based design and simulation while designing a selected computation system. Prerequisites: CSE 361S and CSE 362M.
Credit 3 units. EN: TU

E81 CSE 463M Digital Integrated Circuit Design and Architecture
This is a project-oriented course on digital VLSI design. The course material focuses on bottom-up design of digital integrated circuits, starting from CMOS transistors, CMOS inverters, combinational circuits and sequential logic designs. Important design aspect of digital integrated circuits such as propagation delay, noise margins and power dissipation are covered in the class, as well as design challenges in submicron technology are addressed. The students design combinational and sequential circuits at various levels of abstraction using state-of-the-art CAD environment provided by Cadence Design Systems. The goal of the class is to design a microprocessor in 0.5 micron technology that can be fabricated by a semiconductor foundry. Prerequisites: CSE 260 and ESE 232.
Credit 3 units. EN: TU

E81 CSE 464M Digital Systems Engineering
Design and characterization of digital circuits, reliable and predictable interconnection of digital features, and information transfer over busses and other connections. Topics include: review of MOSFET operation; CMOS logic gate electrical characteristics; system and single-point noise margin and noise
budgets; figures of merit for noise-margin and power-delay product, and trade-off between noise margin and propagation delay; transmission-line driving including reflection, termination, nonzero transition time, lumped and distributed capacitance loads, nonlinear terminations, and applicable conditions for lumped approximations; coupled transmission lines, forward and backward crosstalk, short line approximations, ground bounce, and simultaneous switching noise; timing, clocking and clock distribution for digital circuits; prediction of metastability error rates and design for acceptable probability of failure. Examples and design exercises using systems and interconnections selected from current Computer Engineering practice such as Rambus, PCI bus, GTL, LVDS and others. Prerequisites: ESE 232 and CSE 362M. Credit 3 units. EN: TU

E81 CSE 465M Digital Systems Laboratory
Procedures for reliable digital design, both combinational and sequential; understanding manufacturers specifications; use of special test equipment; characteristics of common SSI, MSI and LSI devices; assembling, testing and simulating design; construction procedures; maintaining signal integrity. Several single-period laboratory exercises, several design projects and application of a microprocessor in digital design. One lecture and one laboratory period a week. Prerequisites: CSE 260M and CSE 361S. Credit 3 units. EN: TU

E81 CSE 467S Embedded Computing Systems
Introduces the issues, challenges and methods for designing embedded computing systems — systems designed to serve a particular application, which incorporate the use of digital processing devices. Examples of embedded systems include PDAs, cellular phones, appliances, game consoles, automobiles and iPod. Emphasis is given to aspects of design that are distinct to embedded systems. The course examines hardware, software and system-level design. Hardware topics include microcontrollers, digital signal processors, memory hierarchy and I/O. Software issues include languages, run-time environments and program analysis. System-level topics include real-time operating systems, scheduling, power management and wireless sensor networks. Students perform a course project on a real wireless sensor network testbed. Prerequisite: CSE 361S. Credit 3 units. EN: TU

E81 CSE 473S Introduction to Computer Networks
A broad overview of computer networking. Topics include layered models of networking protocols, basics of physical layer, data link layer, flow control, error control; local area networks, e.g., Ethernet; wireless networks, IEEE 802.11 (WiFi), cellular wireless networks; Internet protocols, transport protocols, routing algorithms; network security, network management, ATM networks and protocols for networking applications, such as World Wide Web, email and file transfer. Prerequisite: CSE 241. Credit 3 units. EN: TU

E81 CSE 497 Senior Project I
Implementation of a substantive project on an individual basis, involving one or more major areas in computer science. Problems pursued under this framework may be predominantly analytical, involving exploration and extension of theoretical structures, or may pivot around the design/development of solutions for particular applications drawn from areas throughout the University and/or community. In either case, the project serves as a focal point for crystallizing the concepts, techniques and methodologies encountered throughout the curriculum. Students intending to take CSE 497-498 must submit a project proposal for approval by the department during the spring semester of the junior year. Prerequisite: senior standing. Credit 3 units.

E81 CSE 498 Senior Project II
Implementation of a substantive project on an individual basis, involving one or more major areas in computer science. Problems pursued under this framework may be predominantly analytical, involving exploration and extension of theoretical structures, or may pivot around the design/development of solutions for particular applications drawn from areas throughout the University and/or community. In either case, the project serves as a focal point for crystallizing the concepts, techniques and methodologies encountered throughout the curriculum. Students intending to take CSE 497-498 must submit a project proposal for approval by the department during the spring semester of the junior year. Prerequisite: senior standing. Credit 3 units.

E81 CSE 499 Undergraduate Honors Thesis
Working closely with a faculty member, the student investigates an original idea (algorithm, model technique, etc.), including a study of its possible implications, its potential application and its relationship to previous related work reported in the literature. Contributions and results from this investigation are synthesized and compiled into a publication-quality research paper presenting the new idea. Prerequisites: a strong academic record and permission of instructor. Credit 3 units.

E81 CSE 500 Independent Study
Credit variable, maximum 3 units.

E81 CSE 501N Programming Concepts and Practice
An introduction to software concepts and implementation, emphasizing problem solving through abstraction and decomposition. Introduces processes and algorithms, procedural abstraction, data abstraction, encapsulation, and object-oriented programming. Recursion, iteration, and simple data structures are covered. Concepts and skills are mastered through programming projects, many of which employ graphics to enhance conceptual understanding. Java, an object-oriented programming language, is the vehicle of exploration. Active-learning sessions are conducted in a studio setting in which students interact with each other and the professor to solve problems collaboratively. Prerequisites: Comfort with algebra and geometry at the high school level is assumed. Patience, good planning, and organization will promote success. This course assumes no prior experience with programming. Enrollment is typically limited to graduate students. This course will generally not count toward undergraduate engineering degree requirements unless a written exception is granted to a student by the student's faculty adviser. Same as CSE 131 Credit 3 units. BU: SCI EN: TU

E81 CSE 502N Fundamentals of Computer Science
Study of fundamental algorithms, data structures and their effective use in a variety of applications. Emphasizes importance of data structure choice and implementation for obtaining the
This course aims to cultivate an entrepreneurial perspective with particular emphasis on information technology-related activities. The course is jointly offered for business and CSE students, allowing for acculturation between these disciplines. In addition to an introductory treatment of business and technology fundamentals, course topics include: business ethics, opportunity assessment, team formation, financing, intellectual property and university technology transfer. The course features significant participant and guest instruction from experienced practitioners. Prerequisites: none. Credit 3 units.

E81 CSE 513T Theory of Artificial Intelligence and Machine Learning
Mathematical foundations for Artificial Intelligence and Machine Learning. An introduction to the PAC-Semantics ("Probably Approximately Correct") as a common semantics for knowledge obtained from learning and declarative sources, and the computational problems underlying the acquisition and processing of such knowledge. We emphasize the design and analysis of efficient algorithms for these problems, and examine for which representations these problems are known or believed to be tractable. Prerequisites: CSE 240, 241 and mathematical maturity. Any of the following related courses may be helpful, but are not required: CSE 511A, 417A, 517A, 441T/541T. Credit 3 units. EN: TU

E81 CSE 514A Data Mining
Many scientific computing problems are, by nature, statistical. Such problems appear in many domains, such as text analysis, data mining on the web, computational biology and various medical applications. Another source of the statistical nature of such problems is the lack of sufficient information of the problem domains as well as the specific problems at hand. What is available for a typical application is usually a set of data from observation or experiments. The main objective of this course is to gain experience of dealing with statistical data analysis
problems by studying various statistical methods that can be used to make sense out of data, by reading and reviewing literature as well as by working on a specific statistical problem in a selected application domain. Prerequisites: CSE 241 and ESE 326 (or Math 3200) or their equivalent, or permission of the instructor.
Credit 3 units. EN: TU

**E81 CSE 515T Bayesian Methods in Machine Learning**
This course covers machine learning from a Bayesian probabilistic perspective. Bayesian probability allows us to model and reason about all types of uncertainty. The result is a powerful, consistent framework for approaching many problems that arise in machine learning, including parameter estimation, model comparison, and decision making. We begin with a high-level introduction to Bayesian inference, then proceed to cover more-advanced topics. These include inference techniques (exact, MAP, sampling methods, the Laplace approximation, etc.), Bayesian decision theory, Bayesian model comparison, Bayesian nonparametrics, and Bayesian optimization. Prerequisites: a familiarity with machine learning (511A, 417A/517A), probability (e.g., ESE 326), and mathematical maturity.
Credit 3 units. EN: TU

**E81 CSE 516A Multi-Agent Systems**
This course introduces the fundamental techniques and concepts needed to study multi-agent systems, in which multiple autonomous entities with different information sets and goals interact. We study algorithmic, mathematical, and game-theoretic foundations, and how these foundations can help us understand and design systems ranging from robot teams to online markets to social computing platforms. Topics covered may include game theory, distributed optimization, multi-agent learning and decision-making, preference elicitation and aggregation, mechanism design, and incentives in social computing systems. Prerequisites: CSE 240 and 241 and ESE 326 (or Math 320) or equivalents, or permission of instructor. Some prior exposure to artificial intelligence, machine learning, game theory, and microeconomics may be helpful, but is not required.
Credit 3 units. EN: TU

**E81 CSE 517A Machine Learning**
The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience. Recently, many successful machine learning applications have been developed, ranging from data-mining programs that learn to detect fraudulent credit card transactions, to information-filtering systems that learn users’ reading preferences, to autonomous vehicles that learn to drive. There have also been important advances in the theory and algorithms that form the foundation of this field. This course provides a broad introduction to the field of machine learning. Prerequisites: CSE 241 and sufficient mathematical maturity (matrix algebra, probability theory/statistics, multivariate calculus). There is no enrollment limit, but the instructor will hold a take-home placement exam (on basic mathematical knowledge) that is due on the first day of class. Only students who pass the placement exam will be enrolled in the course. Students interested in preparing for the exam are advised to work through the first three weeks of Andrew Ng’s online course on machine learning (https://www.coursera.org/course/ml).
Credit 3 units. EN: TU

**E81 CSE 519T Advanced Machine Learning**
This course provides a close look at advanced machine learning algorithms—their theoretical guarantees (computational learning theory) and tricks to make them work in practice. In addition, this course focuses on more specialized learning settings, including unsupervised learning, semi-supervised learning, domain adaptation, multitask learning, structured prediction, metric learning and learning of data representations. Learning approaches may include graphical models, nonparametric Bayesian statistics, and technical topics such as sampling, approximate inference and nonlinear function optimization. Mathematical maturity and general familiarity of machine learning is required. Prerequisites: CSE 517A or 511A or 7100 and approval by instructor.
Credit 3 units. EN: TU

**E81 CSE 520S Real-Time Systems**
This course covers software technologies for real-time systems and networking such as distributed multimedia, telecommunication management, automobiles, avionics and smart manufacturing. Topics include real-time scheduling, distributed embedded middleware, adaptive performance management and real-time wireless sensor networks. Prior knowledge on embedded and real-time systems is not required. Prerequisite: CSE 422S or equivalent.
Credit 3 units. EN: TU

**E81 CSE 521S Wireless Sensor Networks**
Dense collections of smart sensors networked to form self-configuring pervasive computing systems provide a basis for a new computing paradigm that challenges many classical approaches to distributed computing. Naming, wireless networking protocols, data management and approaches to dependability, real-time, security and middleware services all fundamentally change when confronted with this new environment. Embedded sensor networks and pervasive computing are among the most exciting research areas with many open research questions. This class studies a large number of research papers that deal with various aspects of wireless sensor networks. Students perform a project on a real wireless sensor network composed of tiny devices each consisting of sensors, a radio transceiver and a microcontroller.
Prerequisite: CSE 422S.
Credit 3 units. EN: TU

**E81 CSE 522S Advanced Operating Systems**
This course covers software technologies for real-time systems and networking such as distributed multimedia, telecommunication management, automobiles, avionics and smart manufacturing. Resource management is covered in detail including CPU scheduling, I/O scheduling, interprocess communication models (message passing, remote procedure call and shared memory); virtualization models and techniques; synchronization models and techniques; and resource allocation strategies. Prerequisites: CSE 422S and significant C/C++ programming experience.
Credit 3 units. EN: TU

**E81 CSE 523S Systems Security**
This course examines the intersection between computer design and information security. While performance and efficiency in digital systems have improved markedly in recent decades, computer security has worsened overall in this time frame. To understand why, we explore the role that design choices play
in the security characteristics of modern computer and network systems. Students use and write software to illustrate mastery of the material. Projects include identifying security vulnerabilities, exploiting vulnerabilities, and detecting and defending against exploits. Prerequisites: CSE 361S or permission of instructor. Credit 3 units. EN: TU

**E81 CSE 528S Software Project Management**

An introduction to the issues and basic methods used in managing software development projects. The course includes factors affecting software projects, lifecycle models, project scheduling, size and staffing, progress tracking, software metrics, managing people, and crisis management. The course includes lectures, hands-on training in selected project management tools, and case studies. In addition, each student plans and manages a simulated software project. The course is designed to familiarize software engineers and computer scientists to the issues and problems involved in managing software projects. Prerequisite: CSE 436S, significant industrial software development, or permission of instructor. Credit 3 units.

**E81 CSE 530A Database Management Systems**

A study of data models and the database management systems that support these data models. The design theory for databases is developed and various tools are utilized to apply the theory. General query languages are studied and techniques for query optimization are investigated. Integrity and security requirements are studied in the context of concurrent operations on a database, where the database may be distributed over one or more locations. The unique requirements for engineering design databases, image databases and long transaction systems are analyzed. Prerequisite: CSE 241. Credit 3 units.

**E81 CSE 531S Theory of Compiling and Language Translation**

Algorithms and intermediate representations for automatic program analysis are examined, with an emphasis on practical methods and efficient engineering of program optimization and transformations. The course includes a thorough treatment of monotone data flow frameworks: a mathematical model in which most optimization problems can be specified and solved. The course primarily covers optimizations that are applicable to any target architecture; however, optimizations specific to parallel, distributed and storage-hierarchical systems also are discussed. Prerequisite: CSE 431S or CSE 425S. Credit 3 units. EN: TU

**E81 CSE 532S Advanced Multiparadigm Software Development**

Intensive focus on advanced design and implementation of distributed object computing (DOC) software. Topics covered include reuse of design patterns and software architectures, and developing representative applications using object-oriented and generic frameworks in C++. Design and implementation based on design patterns and frameworks are central themes to enable the construction of reusable, extensible, efficient and maintainable DOC software. Prerequisites: CSE 332S or graduate standing; familiarity with C++, CSE 432S, and CSE 422S or CSE 522S. Credit 3 units. EN: TU

**E81 CSE 535T Programming Language Theory**

This course presents the theoretical foundations of programming languages, using formal techniques. We study how to define programming languages in a formal way, and how to prove meta-theoretic properties about them. Type theory, including powerful typing constructs such as polymorphic and recursive types, receives particular attention. The work for the course includes theoretical exercises as well as a project in which students implement selected aspects of advanced programming languages. Prerequisites: CSE 240 and CSE 241. Credit 3 units.

**E81 CSE 536S Distributed System Design: Models and Languages**

Modern computing environments are highly distributed. This has been the result of major advances in networking technology and their rapid assimilation by a society that functions in a highly distributed and decentralized manner. The goal of this course is to familiarize students with basic concepts, models and languages that shaped recent developments in distributed computing. The focus is on exploring new ways of thinking about computing and communication that made the development of distributed software systems possible. Competing concepts and design strategies are examined both from a theoretical and a practical perspective. Prerequisite: CSE 240 and CSE 241. Credit 3 units. EN: TU

**E81 CSE 538T Modeling and Performance Evaluation of Interconnected Computer Systems**

The increasing pervasiveness of networks, from chip-level to Internet-scale, means that modern computer systems are designed and built from interconnected components. Understanding the design trade-offs this involves and predicting performance under different operating conditions calls for rapidly and effectively evaluating the performance of multiple design combinations. The course reviews a number of analytical tools of relevance in this context (from queueing theory to algorithms to optimization, and possibly game theory if time permits). It explores their use in modeling resources sharing in interconnected systems and estimating various performance metrics of relevance. Prerequisite: basic knowledge of probability theory. Credit 3 units. EN: TU

**E81 CSE 539S Concepts in Multicore Computing**

Nowadays, the vast majority of computer systems are built using multicore processor chips. This fundamental shift in hardware design impacts all areas of computer science — one must write parallel programs in order to unlock the computational power provided by modern hardware. The goal of this course is to study concepts in multicore computing. We examine the implications of the multicore hardware design, discuss challenges in writing high performance software, and study emerging technologies relevant to developing software for multicore systems. Topics include memory hierarchy, cache coherence protocol, memory models, scheduling, high-level parallel language models, concurrent programming (synchronization and concurrent data structures), algorithms for debugging parallel software, and performance analysis. Prerequisites: CSE 241 and either CSE 332S or CSE 361S and being comfortable with writing efficient programs in C/C++.

Credit 3 units. EN: TU
E81 CSE 541T Advanced Algorithms
Provides a broad coverage of fundamental algorithm design techniques with the focus on developing efficient algorithms for solving combinatorial and optimization problems. The topics covered include: greedy algorithms, dynamic programming, linear programming, NP-completeness, approximation algorithms, lower bound techniques, and on-line algorithms. Throughout this course there is an emphasis on correctness proofs and the ability to apply the techniques taught to design efficient algorithms for problems from a wide variety of application areas. Prerequisite: CSE 241. Credit 3 units.

E81 CSE 542T Advanced Data Structures and Algorithms
This course is concerned with the design and analysis of efficient algorithms, focusing principally on algorithms for combinatorial optimization problems. A key element in the course is the role of data structures in algorithm design and the use of amortized complexity analysis to determine how data structures affect performance. The course is organized around a set of core problems and algorithms, including the classical network optimization algorithms, as well as newer and more efficient algorithms. This core is supplemented by algorithms selected from the recent technical literature. Prerequisite: CSE 241. Credit 3 units.

E81 CSE 543T Algorithms for Nonlinear Optimization
The course provides an in-depth coverage of modern algorithms for the numerical solution of multidimensional optimization problems. Unconstrained optimization techniques including gradient methods, Newton’s methods, Quasi-Newton methods and conjugate methods are introduced. The emphasis is on constrained optimization techniques: Lagrange theory, Lagrangian methods, penalty methods, sequential quadratic programming, primal-dual methods, duality theory, nondifferentiable dual methods and decomposition methods. The course also discusses applications in engineering systems and use of state-of-the-art computer codes. Special topics may include large-scale systems, parallel optimization and convex optimization. Credit 3 units.

E81 CSE 544T Special Topics in Computer Science Theory
Cake-cutting algorithms consider the division of resources among a set of participants such that the recipients believe they have been treated fairly. In some cases, a given resource can be divided without loss of value, while in other cases, dividing a resource may lessen its value, perhaps significantly. Notions of fairness include proportionality, envy-freeness and equitability. This course is organized around a rich set of fair-division problems, studying the correctness, complexity, and applicability of algorithms for solving such problems. The problems and algorithms studied span millennia and include recent technical literature. Credit 3 units.

E81 CSE 546T Computational Geometry
Computational geometry is the algorithmic study of problems that involve geometric shapes such as points, lines and polygons. Such problems appear in computer graphics, vision, robotics, animation, visualization, molecular biology and geographic information systems. This course covers data structures that are unique to geometric computing, such as convex hull, Voronoi diagram, Delaunay triangulation, arrangement, range searching, KD-trees and segment trees. Also covered are algorithms for polygon triangulation, shortest paths, the post office problem and the art gallery problem. Prerequisite: CSE 241. Credit 3 units.

E81 CSE 547T Introduction to Formal Languages and Automata
An introduction to the theory of computation, with emphasis on the relationship between formal models of computation and the computational problems solvable by those models. Specifically, this course covers finite automata and regular languages; Turing machines and computability; and basic measures of computational complexity and the corresponding complexity classes. Prerequisite: CSE 240. Credit 3 units.

E81 CSE 548T Concurrent Systems: Design and Verification
Formerly CSE 563T. Concurrency presents programmers with unprecedented complexity further exacerbated by our limited ability to reason about concurrent computations. Yet, concurrent algorithms are central to the development of software executing on modern multiprocessors or across computer networks. This course reviews several important classes of concurrent algorithms and presents a formal method for specifying, reasoning about, verifying and deriving concurrent algorithms. The selected algorithms are judged to have made significant contributions to our understanding of concurrency. Rigorous treatment of the design and programming process is emphasized. Students entering this course must be familiar with predicate calculus and sequential algorithms. Upon completion of this course students are able to reason completely formally about small concurrent programs and to apply systematically and correctly their formal skills to larger problems. Prerequisites: CSE 240 and CSE 241. Credit 3 units.

E81 CSE 549T Theory of Parallel Systems
The course covers parallel programming models, languages and algorithms. We cover both classic and recent results in parallel computing. The list of topics includes scheduling theory, parallel algorithms (in various models such as PRAM, log P, etc.), cache-coherence protocols and external memory algorithms. The focus is on design and performance analysis of algorithms. Credit 3 units.

E81 CSE 550A Mobile Robotics
An introduction to the design and implementation of intelligent mobile robot systems. This course covers the fundamental elements of mobile robot systems from a computational standpoint. Issues such as software control architectures, sensor interpretation, map building and navigation are covered, drawing from current research in the field. Students also design and build a small mobile robot and program it to perform simple tasks in real-world environments. Prerequisites: CSE 241 and either ESE 326 or Math 3200. Credit 3 units. EN: TU

E81 CSE 550S Mobile Robotics
An introduction to the design and implementation of intelligent mobile robot systems. This course covers the fundamental elements of mobile robot systems from a computational standpoint. Issues such as software control architectures, sensor
interpretation, map building and navigation are covered, drawing from current research in the field. Students also design and build a small mobile robot and program it to perform simple tasks in real-world environments. Prerequisites: CSE 241 and either ESE 326 or Math 320.
Credit 3 units. EN: TU

E81 CSE 552A Advanced Computer Graphics
This course covers advanced topics in graphics in the areas of modeling, rendering, volume rendering, image-based rendering and image processing. Topics include, but are not limited to, subdivision surfaces, splines, mesh simplification, implicit or blobby modeling, radiosity, procedural textures, filtering, BRDFs and procedural modeling. The class has several structured programming assignments and an optional final group project. Students are exposed to the wide variety of techniques available in graphics and also pick one area to study in depth. Prerequisites: CSE 332S and CSE 452A.
Credit 3 units. EN: TU

E81 CSE 553S Advanced Mobile Robotics
This course covers advanced topics from the theory and practice of mobile robotics. Students read, present and discuss papers from the current research literature. There is a substantial programming project, in which students implement and test ideas from the current research literature on one of the department's research robot platforms. Prerequisites: CSE 550A and strong programming skills (preferably in C++).
Credit 3 units. EN: TU

E81 CSE 554A Geometric Computing for Biomedicine
With the advance of imaging technologies deployed in medicine, engineering and science, there is a rapidly increasing amount of spatial data sets (images, volumes, point clouds, etc.) that need to be processed, visualized and analyzed. This course focuses on a number of geometry-related computing problems that are essential in the knowledge discovery process in various spatial-data-driven biomedical applications. These problems include visualization, segmentation, mesh construction and processing, shape representation and analysis. The course consists of lectures that cover theories and algorithms, and a series of hands-on programming projects using real-world data collected by various imaging techniques (CT, MRI, electron cryo-microscopy, etc.). Prerequisites: CSE 332 and CSE 452, or approval by instructor.
Credit 3 units. EN: TU

E81 CSE 555A Computational Photography
Computational Photography describes the convergence of computer graphics, computer vision, and the Internet with photography. Its goal is to overcome the limitations of traditional photography using computational techniques to enhance the way we capture, manipulate and interact with visual media. In this course we study many interesting, recent image-based algorithms and implement them to the degree that is possible. Topics may include: cameras and image formation, human visual perception, image processing (filtering, pyramids), image blending and compositing, image retargeting, texture synthesis and transfer, image completion/inpainting, super-resolution, deblurring, denoising, image-based lighting and rendering, high dynamic range, depth and defocus, flash/no flash photography, coded aperture photography, single/multiview reconstruction, photo quality assessment, non photorealistic rendering, modeling and synthesis using Internet data, and others. Prerequisites: CSE 452A, CSE 554A, or CSE 559A. Permission of instructor required to enroll.
Credit 3 units. EN: TU

E81 CSE 556A Human-Computer Interaction Methods
This course is designed to introduce tools and methods from Human-Computer Interaction that enable you to create effective user interfaces. We cover techniques that can be used at different stages in the software development cycle and techniques that can be used both with and without involving users. We study how to maximize interface usability and efficiency as well as how to design for, and measure, things like fun and persuasiveness. You gain experience applying HCI techniques through a group design, development and evaluation project. Prerequisites: CSE 132 and at least one additional systems course. Students who enroll in this course are expected to be comfortable with the process of building user interfaces using Java JFC/Swing.
Credit 3 units. EN: TU

E81 CSE 558A Motion Planning
This course studies the general motion planning problem: computing a sequence of motions that transforms a given (initial) arrangement of physical objects to another (goal) arrangement of those objects. Many motion planning methods were developed in the realm of robotics research. For example, a typical problem might be to find a sequence of motions (called a path) to move a robot from one position to another without colliding with any objects in its workspace. However, the general motion planning problem studied arises in many other application domains as well. For example, assembly planning (e.g., finding a valid order for adding the parts when building an engine), mechanical CAD studies (e.g., can you remove a certain part from an engine without taking the engine apart), artificial life simulations (e.g., moving a herd of animals from one location to another), and medicine (e.g., can a drug molecule reach a protein molecule). Prerequisite: CSE 241.
Credit 3 units. EN: TU

E81 CSE 559A Computer Vision
Computer vision is the process of automatically extracting information from images and video. This course covers imaging geometry (camera calibration, stereo and panoramic image stitching) and algorithms for video surveillance (motion detection and tracking), segmentation and object recognition. Final projects for the course explore challenges in analysis of real-world data. Students with nonstandard backgrounds (such as video art, or the use of imaging in physics and biology) are encouraged to contact the instructor. Prerequisites: CSE 241 and linear algebra.
Credit 3 units. EN: TU

E81 CSE 560M Computer Systems Architecture I
An exploration of the central issues in computer architecture: instruction set design, addressing and register set design, control unit design, microprogramming, memory hierarchies (cache and main memories, mass storage, virtual memory), pipelining, bus organization, Reduced Instruction Set Computers (RISC), and Complex Instruction Set Computers (CISC). Architecture design and evaluation using VHDL and/or instruction set simulation. Prerequisites: CSE 361S and CSE 260M.
Credit 3 units. EN: TU
E81 CSE 561M Computer Systems Architecture II
Advanced techniques in computer system design. Selected topics from: processor and system-on-chip design (multicore organization, system-level integration), run-time systems, memory systems (topics in locality and special-purpose memories), I/O subsystems and devices, systems security, and power considerations. Prerequisite: CSE 560M or permission of instructor.
Credit 3 units. EN: TU

E81 CSE 564M Advanced Digital Systems Engineering
This course focuses on advance sensor design. The class covers various basic analog and digital building blocks that are common in most sensor integrated circuits. The class extensively uses state-of-the-art CAD program Cadence to simulate and analyze various circuit blocks. The first half of the course focuses on analyzing various operational amplifiers, analog filters, analog memory and analog to digital converters. The second half of the course focuses on understanding the basic building blocks of imaging sensors. The class has a final project consisting of designing a smart sensor using Cadence tools. Prerequisites: ESE 232 and CSE 362M.
Credit 3 units. EN: TU

E81 CSE 565M Acceleration of Algorithms in Reconfigurable Logic
Reconfigurable logic, in the form of Field-Programmable Gate Arrays (FPGAs), enables the deployment of custom hardware for individual applications. To exploit this capability, the application developer is required to specify the design at the register-transfer level. This course explores techniques for designing algorithms that are amenable to hardware acceleration as well as provides experience in actual implementation. Example applications are drawn from a variety of fields, such as networking, computational biology, etc. Prerequisites: Basic digital logic (CSE 280M) and some experience with a hardware description language (e.g., VHDL or Verilog).
Credit 3 units. EN: TU

E81 CSE 566M High-Performance Computer Systems
Many applications make substantial performance demands upon the computer systems upon which those applications are deployed. In this context, performance is frequently multidimensional, including resource efficiency, power, execution speed (which can be quantified via elapsed run time, data throughput or latency), etc. Modern computing platforms exploit parallelism and architectural diversity (e.g., co-processors such as graphics engines and/or reconfigurable logic) to achieve the desired performance goals. This course addresses the practical aspects of achieving high performance on modern computing platforms. This includes questions ranging from how the computing platform is designed to how are applications and algorithms expressed to exploit the platform's properties. Particular attention is given to the role of application development tools. Prerequisite: graduate standing or permission of instructor.
Credit 3 units.

E81 CSE 567M Computer Systems Analysis
Comparing systems using measurement, simulation and queueing models. Common mistakes and how to avoid them, selection of techniques and metrics, art of data presentation, summarizing measured data, comparing systems using sample data, introduction to experimental design, fractional factorial designs, introduction to simulation, common mistakes in simulations, analysis of simulation results, random number generation, random variate generation, commonly used distributions, introduction to queuing theory, single queues and queueing networks. The techniques of the course can be used to analyze and compare any type of systems including algorithms, protocols, network or database systems. Students do a project involving application of these techniques to a problem of their interest. Prerequisites: CSE 131 and CSE 260M.
Credit 3 units. EN: TU

E81 CSE 568M Imaging Sensors
This course covers topics on digital imaging sensors including basic operations of silicon photodetectors; CCD and CMOS passive and active sensor operation; temporal and spatial noise in CMOS sensors; spatial resolution and MTF; SNR and dynamic range; high dynamic range architectures and application specific imaging sensors such as polarization imaging and fluorescent imaging sensors.
Credit 3 units.

E81 CSE 569M Parallel Architectures and Algorithms
A number of contemporary parallel computer architectures are reviewed and compared. The problems of process synchronization and load balancing in parallel systems are studied. Several selected applications problems are investigated and parallel algorithms for their solution are considered. Selected parallel algorithms are implemented in both a shared memory and distributed memory parallel programming environment. Prerequisites: graduate standing and knowledge of the C programming language.
Credit 3 units. EN: TU

E81 CSE 570S Recent Advances in Networking
This course covers the latest advances in networking. The topics include Networking Trends, Data Center Network Topologies, Data Center Ethernet, Server Virtualization, Storage Virtualization, Carrier Ethernet, Application Delivery Networking, Virtual Bridging, Big Data, Networking Issues for Big Data, LAN Extension and Virtualization using Layer 3 Protocols, Multi-Tenant Isolation and Network Virtualization in Cloud Data Centers, OpenFlow, OpenFlow Controllers and Tools, Software Defined Networking (SDN), Network Function Virtualization (NFV), Internet of Things (IoT), Networking Protocols for IoT. Prerequisite: CSE 473 or equivalent.
Credit 3 units.

E81 CSE 571S Network Security
A comprehensive treatment of network security. Topics include virus, root kits, malware, TCP/IP security attacks, operating systems security, secret key cryptography, modes of operation, hashes and message digests, public key algorithms, authentication systems, Kerberos, Public Key Infrastructures (PKI), IPsec, Internet Key Exchange (IKE), Secure Socket Layer (SSL), Transport Layer Security (TLS), e-mail security, virtual private networks, authentication, authorization, accounting (AAA), WiFi security, Domain Name System security (DNSSEC), intrusion detection systems. Prerequisite: CSE 473S.
Credit 3 units. EN: TU

E81 CSE 573S Protocols for Computer Networks
An introduction to the design, performance analysis and implementation of existing and emerging computer network protocols. Topics include networking, operating systems, application programming interfaces, transport, and network-layer protocols. Prerequisites: CSE 361M and CSE 472. Credit 3 units.
protocols. Protocols include multiple access protocols (e.g., CSMA/CD, token ring), internet, working with the internet protocol (IP), transport protocols (e.g., UDP, TCP), high-speed bulk transfer protocols, and routing protocols (e.g., BGP, OSPF). General topics include error control, flow control, packet switching, mechanisms for reliable, ordered and bounded-time packet delivery, host-network interfacing and protocol implementation models. Substantial programming exercises supplement lecture topics. Prerequisite: CSE 473S or permission of the instructor.
Credit 3 units. EN: TU

E81 CSE 574S Wireless and Mobile Networking
First course in wireless networking providing a comprehensive treatment of wireless data and telecommunication networks. Topics include wireless networking trends, key wireless physical layer concepts, wireless local area networks, wireless personal area networks, WiMAX (physical layer, media access control, mobility and networking), IEEE 802.22 wireless regional area networks, IEEE 802.21 media independent handover, wireless cellular networks: 1G and 2G, 2.5G, 3G and 4G, Mobile IPv4, Mobile IPv6, TCP over wireless networks, ad hoc networks — issues and routing, wireless sensor networks, wireless mesh and multi-hop relay networks, radio frequency identification (RFID). Prerequisites: CSE 473S or permission of the instructor.
Credit 3 units. EN: TU

E81 CSE 577M Design and Analysis of Switching Systems
Switching is a core technology in a wide variety of communication networks, including the Internet, circuit-switched telephone networks and optical fiber transmission networks. The last decade has been a time of rapid development for switching technology in the Internet. Backbone routers with 10 Gb/s links and aggregate capacities of hundreds of gigabits per second are becoming common, and advances in technology are now making multi-terabit routers practical. This course is concerned with the design of practical switching systems and evaluation of their performance and complexity. Prerequisites: CSE 241, 473S and ESE 326.
Credit 3 units. EN: TU

E81 CSE 583A Topics in Computational Molecular Biology
Formerly CSE 543T. In-depth discussion of problems and methods in Computational Molecular biology. Each year three topics are covered and these change yearly. Prerequisite: Biol 5495 or instructor's consent.
Same as Biol 5497
Credit 2 units.

E81 CSE 584A Algorithms for Biosequence Comparison
This course surveys fundamental algorithms for comparing and organizing biological sequences. Emphasis is placed on techniques that are useful for implementing biosequence databases and comparing long sequences, such as entire genomes. Many of these techniques are also of interest for more general string processing and for building and mining of textual databases. Algorithms are presented rigorously, including proofs of correctness and running time where feasible. Topics include classical string matching, suffix trees, exclusion methods, multiple alignments and the design of BLAST and related biosequence comparison tools. Students complete written assignments and implement advanced comparison algorithms to address problems in bioinformatics. This course does not require a biology background. Prerequisites: CSE 241, graduate standing or permission of instructor.
Credit 3 units. EN: TU

E81 CSE 587A Algorithms for Computational Biology
This course focuses on how to sequence and analyze a genome, emphasizing computational and algorithmic issues. After taking this course, you should be able to parachute into a genome informatics group, understand what's going on, and do something useful on your first day. Topics covered include: the essential biology, the essential probability theory, base calling and quality clipping, genome assembly (including aspects of sequence alignment), predicting protein-coding genes (including Hidden Markov Models and comparative genomics approaches), predicting gene function by comparing to proteins of known function, and advanced topics in sequence alignment. This course includes a combination of paper-and-pencil homework assignments and programming labs in C. Prerequisites: CSE 241 or CSE 502N or Biol 5495.
Credit 3 units.