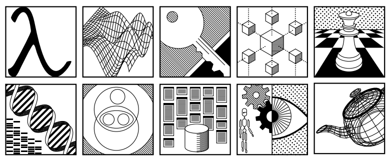
UNDERGRADUATE HANDBOOK

2013–2014 Edition



Department of Computer Science
Yale University

Contents

1 Introduction

The Department of Computer Science at Yale University was started in 1969 as a small graduate department and began offering an undergraduate major in 1972. There are now 20 regular faculty members, 1 emeritus professor, 1 lecturer, 6 affiliated faculty, and 3 research scientists; more than 100 undergraduate majors (38 seniors last year); and more than 50 graduate students. The department offers more than 30 different undergraduate courses each year, all taught by faculty. Further information is available from the departmental web page `http://www.cs.yale.edu`.

2 Faculty

Department Chair
Holly Rushmeier

Director of Undergraduate Studies
Stanley Eisenstat

Director of Graduate Studies
Vladimir Rokhlin

Professors

Dana Angluin	David Gelernter	Martin Schultz (emeritus)
James Aspnes	Paul Hudak	Zhong Shao
Julie Dorsey	Drew McDermott	Avi Silberschatz
Stanley Eisenstat	Vladimir Rokhlin	Daniel Spielman
Joan Feigenbaum	Holly Rushmeier	Yang Richard Yang
Michael Fischer	Brian Scassellati	Steven Zucker

Associate Professors
Daniel Abadi

Assistant Professor

Bryan Ford	Ruzica Piskac

Lecturer
Brad Rosen

Affiliated Faculty

Dirk Bergemann	Mark Gerstein	Frederick Shic
Ronald Coifman	Gil Kalai	Sekhar Tatikonda

Research Scientists

Rob Bjornson	Nicholas Carriero	Andrew Sherman

3 Overview of the Department

Computer science is one of the most dynamic and fertile intellectual enterprises of our age. At Yale our focus is on the structure, design, and fundamental properties of computers and computer programs and on methods for using computers to solve significant problems. We use mathematics extensively in the design and analysis of problem-solving techniques and the exploration of fundamental properties of computation, and draw heavily on techniques from engineering and from the natural sciences as well.

There are six main areas of study: artificial intelligence, computer graphics, computer systems, programming languages, scientific computing, and theory of computation. In addition there are collaborations with other disciplines, including economics, engineering, law, linguistics, mathematics, medicine, psychology, and the arts.

3.1 Artificial Intelligence

Artificial Intelligence is the study of computational models of the mind. At Yale there is a wide variety of topics studied, including vision, robotics, planning, learning, and computational neuroscience.

The term “artificial intelligence” is somewhat misleading because the focus of research in the field is often on more mundane activities, such as simple visual perception, than the word “intelligence” would suggest. The field has learned over the years that the effortlessness of a skill such as vision is deceptive, that in fact the brain does a great deal of hard labor behind the scenes to allow us to see without conscious effort. It will take us years to duplicate the skills that nature evolved over eons.

In general we think it is a mistake for AI research to focus on central mental function and ignore input and output. In the long run machines will not be treated as intelligent unless they can perceive and manipulate the objects around them. Real perception and action impose stubborn constraints on thinking. Sophisticated robot planning is wasted if the robot crashes into the wall while trying to generate a predicate-calculus description of the world in front of it. Hence our focus is on real-time perceptual control of behavior, in both natural and artificial systems.

AI uses many of the same techniques as other areas of computer science application, from numerical optimization to symbolic indexing. The key to solving any problem is always the algorithm and its analysis. The goal is always to characterize precisely a set of problems and demonstrate an algorithm that solves them with reasonable efficiency. But, at least at its current state of development, AI is of necessity more exploratory than other areas. We are often forced to define a problem at the same time that we try to solve it. It often happens that we don't know how to analyze the performance of an algorithm with existing tools, but we believe that its average-case performance is much better than its worst-case performance, and this belief must be backed up with experiments.

Faculty members working in this area are Drew McDermott, Brian Scassellati, and Steven Zucker.

3.2 Computer Graphics

Research in computer graphics at Yale includes sketching and alternative design techniques, material and texture models, the role of models of human perception in computer graphics, and recovering shape and reflectance from images. Applications that drive this work are architectural design, cultural heritage documentation and analysis, and the study of biological forms.

Computer graphics is used extensively in a wide range of domains—from feature film and games to medical visualization and financial analysis. However impressive the growth of computer graphics applications has been over the past forty years, the goal of easily authoring computer graphics models input remains elusive. At Yale, the research in modeling includes sketching systems for early conceptual design and the capture and editing of digital models of existing physical objects at a range of scales—from entire buildings to individual objects.

Computer graphics models need to include material appearance properties as well as geometry. Unfortunately, the models widely used in computer graphics assume that the materials are both pristine and immutable, even though real materials are neither. The goal of research on material and texture models at Yale is to devise new material representations and expressive interfaces for editing such representations, to develop novel methods to simulate materials and the processes that affect their appearance, and to physically measure the input required for material models.

Faculty members working in this area are Julie Dorsey and Holly Rushmeier.

3.3 Computer Systems

Computer systems research at Yale is divided into three sub-areas (with a large degree of overlap and collaboration across these sub-areas): database systems, operating systems, and networking systems.

Database systems provide an environment for storage and retrieval of both structured and semi-structured data. Such systems were originally designed for use in business-type applications. Today, however, they are being utilized in many other application domains, including scientific computing, networking, and bioinformatics. Research topics at Yale include transaction management, data warehousing, Web-scale databases, real-time systems, multimedia systems, approximate queries, and data mining.

The role of operating systems has evolved over time, from sharing one device's resources among many users in the mainframe era, to providing convenient user interface, storage, and networking abstractions in the personal computer era. As we transition to the ubiquitous computing era, operating systems must now manage a user's information and computation across many computers and devices. Yale is developing new operating system architectures, application environments, and security frameworks to meet today's challenges across the computing spectrum, from mobile personal devices to large-scale Internet services built on grids of many-core processors.

Computer networks allow computers to communicate with one another, and of course form the backbone of the Internet. But although they have become a critical infrastructure of our information-based society, they still have not achieved the reliability of traditional telephone networks. Research at Yale concentrates on designing highly robust and efficient Internet backbone networks, by combining computer science with optimization, economics, and game theory. Peer-to-peer (P2P) is emerging as a new paradigm for network application development, as witnessed by the wide usage of P2P file-sharing and video-streaming applications. However, these applications not only generate a large volume of traffic, but also may unnecessarily spread traffic across the whole Internet, leading to inefficiency. Research at Yale focuses on designing effective architecture and algorithms to improve both application performance and Internet operation efficiency.

Faculty members in the Computer Systems and Networking area are Daniel Abadi, Bryan Ford, Avi Silberschatz, and Yang Richard Yang.

3.4 Programming Languages

Programming languages are the main vehicle for man-machine communication. They provide a way to express an algorithm as a program and impact the way we think of a computer system. Several languages developed at or associated with the department (in particular Haskell, ML, and Linda) have achieved worldwide currency, reflecting the department's leadership in the areas of functional programming and parallel computing. Parallel languages such as Linda are tools for building programs that do many things simultaneously; functional languages such as Haskell and ML provide a mathematical approach to programming based on a view of a program as a set of simple equations. Applications of programming language research include graphics and animation, networking, computer music, robotics, graphical user interfaces, and systems programming.

Three particular areas of study are *formal methods, program synthesis,* and *software ensembles*. Formal methods emphasize the use of formal mathematics to ensure the correctness, reliability, and maintainability of complex software systems. At Yale the study of formal methods focuses on functional programming and related ideas such as computational logic, denotational semantics, type theory, category theory, program transformation, domain specific languages, and high-level abstraction techniques.

The goal of software synthesis is to automatically generate code that satisfies a given specification. Code obtained this way is correct by construction. The specification can be explicitly given, or it can be derived from context. Synthesis research at Yale is mainly focused on how to use formal methods, e.g., decision procedures, for automatic code construction.

Software ensembles are programs that are built out of many separate, coordinated activities, with an emphasis on recognizing and understanding the properties that all such systems share. The search for a precise definition of "coordination language," and the development of internet applications are new focuses of the software ensemble project.

Faculty working in this area are David Gelernter, Paul Hudak, Ruzica Piskac, and Zhong Shao. Nick Carriero is a research scientist.

3.5 Scientific Computing

During the past forty years computers have dramatically changed engineering, medicine, and science by making it possible to test designs and run trials without first building a prototype for each product or conducting an elaborate experiment for each trial. The impact of this new ability, this power to simulate the real thing, has placed scientific computation as the keystone between theory and applications.

Scientific computing uses concepts and methodologies from numerical linear and nonlinear algebra and boundary value problems for differential equations. In addressing these areas, research at Yale emphasizes algorithm development, theoretical analysis, systems modeling, and programming considerations. Algorithm development is concerned with finding new, fast, and/or parallel methods. Theoretical analysis investigates rates of convergence, stability, optimality, and operation counts. Systems modeling examines the performance implications of the interactions between computationally intensive algorithms, operating systems, and multiprocessors. Programming considerations include coding efficiency, numerical accuracy, generality of application, data structures, and machine independence.

Faculty working in this area are Stanley Eisenstat, Vladimir Rokhlin, and Martin Schultz. Rob Bjornson and Andrew Sherman are research scientists.

3.6 Theory of Computation

Theory of computation involves the use of powerful mathematical tools to obtain deep insights into fundamental problems of computation. Not being constrained by the current state of technology, research in the area explores "what is imaginable" as well as "what is."

At Yale theory research is concentrated in the areas of algorithms, distributed computing, economics and computation, graphs and networks, machine learning, and security. Algorithms involves the invention and analysis of algorithms for sequential and parallel models of computation; fundamental research in the areas of machine learning and graphs and networks is largely motivated by the search for efficient algorithms. Distributed computing extends the domain of computation to encompass systems of concurrent communicating asynchronous processors such as peer-to-peer networks and cluster computers. Uncertainty and failures inherent in such systems provide a special focus for research in this area. Economics and computation studies models that combine concepts from economics, such as incentives, social welfare, and game theory, with concerns from computing, such as efficient use of computational resources. Research in security concerns the reliability, privacy, and availability properties of information systems; it is closely related to areas such as algorithms, complexity theory, cryptography, and randomness.

Two concepts that are fundamental to all areas of computer science are computing devices and algorithms. A computing device may be a computer, a network of computers, a circuit, a robot, or a software simulator or interpreter. An algorithm is a precise description of how some task is to be executed by a computing device. The curriculum in theory of computation is designed to provide a solid theoretical basis for the understanding of computing devices and algorithms.

The most important tool for this kind of theoretical understanding is "appropriate abstraction." The idea is to make a theoretical model of (for example) a computer that ignores enough of the details of any specific computer to be general, but is still specific enough that its properties give useful insight into the capabilities of actual computers. The ability to use various levels of abstraction, from the immediately practical to the quite theoretical, is a lasting benefit of an education in computer science.

There is considerable contact with discrete mathematics, graph theory, number theory, mathematical logic, probability and statistics, operations research, economics, computational finance, and other related areas of study.

Faculty working in this area are Dana Angluin, James Aspnes, Joan Feigenbaum, Michael Fischer, and Daniel Spielman.

4 Computing Facilities

The faculty, researchers, and students within the department use a variety of computing resources, ranging from conventional PC's and scientific workstations to high-powered compute-servers and workstation clusters used as parallel computers. These systems are interconnected by an Ethernet local area network, which is in turn connected to the Internet via fiber optic technology to the campus backbone.

The educational computing facility is open to students in computer science courses for classwork and to undergraduate majors for unsponsored research and other academic purposes. Affectionately known as the "Zoo" (web site: `http://zoo.cs.yale.edu`) and the site of regular late-night pizza parties, it is located on the third floor of Arthur K. Watson Hall, which houses the department. Its 35 Linux workstations and 3 Windows 7 graphics workstations have quad-core, Intel Xeon E5-1620 processors, 32 Gigabytes of RAM, and 27-inch flat panel monitors. This facility can be accessed locally and remotely 24 hours a day, 365 days a year.

5 Degree Programs in Computer Science

The computer science curriculum offers students training in the theory and practice of computing. A major in computer science prepares one for a job in the field or for graduate study leading to teaching or research. A computer science undergraduate education followed by graduate study in law, business, or medicine is another strong combination.

The department offers both a Bachelor of Science and a Bachelor of Arts major in Computer Science (see §5.1) and a combined B.S. / M.S. program (see §5.3). It also participates in joint majors with the Departments of Mathematics (see §5.4), Psychology (see §5.5), and Electrical Engineering (see §5.6).

Each of these programs provides a solid technical education yet allows students either to take the broad range of courses in other disciplines that is an essential part of a liberal education or to complete the requirements of a second major.[1] Thus the number of courses required is somewhat less than at other schools.

The programs are built around a common core of five computer science courses. The first, CPSC 201 *Introduction to Computer Science*, is a survey that illustrates the breadth and depth of the field to students who have already completed a one-term introductory course in programming. The others cover

[1]Roughly 25% of our students complete a second major such as Economics, Music, Political Science, or Theater Studies.

discrete mathematics; data structures; systems programming and computer architecture; and algorithm analysis and design. Together they include the material that every student of computer science should know.

This core is supplemented by a set of electives (and for the joint majors, a set of core courses in the other discipline). The electives give students great flexibility in tailoring the program to specialize in particular areas of computer science or to broaden their knowledge in a variety of areas.

The capstone of each program is the senior project, which lets students experience the challenges and rewards of original scientific research under the guidance of a faculty member. These projects deal with problems that cross the boundaries between courses and can involve complex and imaginative use of computers.

5.1 B.S. and B.A. in Computer Science

A student can earn either a Bachelor of Science or a Bachelor of Arts degree in Computer Science. The B.S. program is designed for students who plan to continue in computing after graduation, including technical management and consulting. The B.A. provides a solid computer science background as preparation for work in other fields.[2]

The B.S. and B.A. degree programs both require the same five core courses

◇ CPSC 201a or b *Introduction to Computer Science*
◇ CPSC 202a *Mathematical Tools for Computer Science*[3]
◇ CPSC 223b *Data Structures and Programming Techniques*
◇ CPSC 323a *Introduction to Systems Programming and Computer Organization*
◇ CPSC 365b *Design and Analysis of Algorithms*

and a senior project (see §5.2) taken as

◇ CPSC 490a or b *Special Projects.*

In addition the B.S. program requires six intermediate or advanced computer science courses as electives, for a total of twelve courses; the B.A., four, for a total of ten. Neither CPSC 480a or b *Directed Reading* nor CPSC 490a or b may be counted as electives.

The prerequisite structure of the core courses is shown in Figure 1. Typical schedules beginning in the freshman and sophomore years are given in Tables 1 and 2.

[2]Students with interests less squarely focused on computer science and extending to the arts (Architecture, Art, History of Art, Music, or Theater Studies) may prefer the major in Computing and the Arts.

[3]Students with the appropriate background are encouraged to substitute MATH 244a *Discrete Mathematics* for CPSC 202a.

Figure 1: The prerequisite structure of the core courses.

$$\begin{tikzcd}
\text{CPSC 201a or b} \arrow[r] & \text{CPSC 223b} \arrow[r] \arrow[dr] & \text{CPSC 323a} \\
\text{CPSC 202a} \arrow[rr] & & \text{CPSC 365b}
\end{tikzcd}$$

Table 1: Sample B.S. programs for a student starting in the freshman year. Omit two electives from either to get a B.A. program.

Fall	*Spring*		*Fall*	*Spring*
CPSC 201	CPSC 223	*Freshman*		CPSC 201
CPSC 202	CPSC 365	*Sophomore*	CPSC 202	CPSC 223
CPSC 323	Elective			Elective
Elective	Elective	*Junior*	CPSC 323	CPSC 365
Elective	Elective		Elective	Elective
CPSC 490	Elective	*Senior*	Elective	CPSC 490
			Elective	Elective

Table 2: Sample B.S. programs for a student starting in the sophomore year. Omit two electives from either to get a B.A. program.

Fall	*Spring*		*Fall*	*Spring*
CPSC 201	CPSC 223	*Sophomore*		CPSC 201
CPSC 202	Elective			
CPSC 323	CPSC 365	*Junior*	CPSC 202	CPSC 223
Elective	Elective		Elective	CPSC 365
				Elective
Elective	CPSC 490	*Senior*	CPSC 323	CPSC 490
Elective	Elective		Elective	Elective
			Elective	Elective

Students are strongly advised to complete CPSC 201a or b and 223b by the end of their sophomore year. Otherwise the choice of electives may be somewhat limited, especially in the B.S. program.

All sophomore, junior, and senior majors should have their programs approved by their class advisor (see §6.1) or the director of undergraduate studies.

All courses counting toward the major must be taken for a letter grade.

Electives

The five core courses cover the material that every student of computer science should know; the electives (see §7.2 and §7.3) give students an opportunity to specialize in particular areas of computer science:

- *Artificial Intelligence*: CPSC 470a, 471b, 472a, 473b, 475a
- *Computer Graphics*: CPSC 478b, 479b
- *Computer Systems*: CPSC 422b, 423a, 426a, 433a, 434a, 437b, 438b
- *Programming Languages*: CPSC 421b, 427a, 428b, 430a, 431b, 432b
- *Scientific Computing*: CPSC 424b, 440b, 445a
- *Theory of Computation*: CPSC 455a, 457a, 462a, 465b, 467a, 468a, 469b.

However, we encourage students to take electives in several different areas.

Students considering graduate study in computer science (either immediately following graduation or after working for some years) are advised to take a programming-intensive course such as

◇ CPSC 421a *Compilers and Interpreters*
◇ CPSC 422b *Operating Systems*

and a course in theoretical computer science such as

◇ CPSC 462a *Graphs and Networks*
◇ CPSC 468a *Computational Complexity*
◇ CPSC 469b *Randomized Algorithms*,

as well as courses in their intended area of study.

Students interested in applications of computers to scientific and engineering problems are advised to take

◇ CPSC 440b *Numerical Computation*

in addition to computational courses in Applied Mathematics and Engineering and Applied Science.

To encourage study in interdisciplinary areas where computer science plays a major role, *advanced* courses[4] in other departments that involve con-

[4]An advanced course is generally one with at least one intermediate course as a prerequisite, and an intermediate course is generally one with at least one (introductory) course as a prerequisite. One exception is MATH 120a or b *Calculus of Functions of Several Variables*, which is considered to be an introductory course.

cepts from computer science and are particularly relevant to an individual program may, with permission of the class advisor or the director of undergraduate studies, be counted as electives. Generally at most two such courses may be used to satisfy the requirements for the B.S. program (one for the B.A. program).

Even if they cannot be counted as electives, some courses in mathematics (e.g., calculus, linear algebra, probability and statistics, optimization, and discrete mathematics) may be beneficial. For example, some graduate programs require calculus and linear algebra.

SUMMARY OF REQUIREMENTS

Prerequisites: None
Number of courses: *B.S. degree*—twelve term courses taken for a letter grade (including the senior project); *B.A. degree*—ten term courses taken for a letter grade (including the senior project)
Specific courses required: *B.S. and B.A. degrees*—CPSC 201a or b, 202a, 223b, 323a, and 365b
Distribution of courses: *B.S. degree*—six additional intermediate or advanced CPSC courses. *B.A. degree*—four additional intermediate or advanced CPSC courses
Substitution permitted: Advanced courses in other departments, with permission of the class advisor or the director of undergraduate studies
Senior requirement: Independent project (CPSC 490a or b)

5.2 Special Projects

CPSC 490a or b *Special Projects* fulfills the senior project requirement of the B.S. and B.A. programs. Students select a faculty advisor to guide them in research in a subfield of computer science and submit a written report on the results. Many of these projects break new ground, and papers with Yale undergraduate authors have been published in leading computer science journals.

5.2.1 Frequently Asked Questions Regarding CPSC 490

Who is the Senior Class Advisor (SCA)?

The Senior Class Advisor (SCA), who administers the CPSC 490 projects, is the faculty member who advises and signs the course schedules of all graduating seniors (i.e., the members of the Class of 2014). This year's SCA is Professor Dana Angluin (email: `dana.angluin@yale.edu`).

What are the deadlines?

Senior majors enrolled in CPSC 490 must submit the CPSC 490 form (which includes a 3-page description of the project and the list of deliverables) by noon on the fourth Thursday of the term. Other students must have this form approved by the DUS at least two days before course schedules are due. Note: Joint majors must submit the CPSC 490 Joint Major form instead.

All students must complete the end-of-term requirements (see below) by noon on the last day of reading period.

Are there any specific requirements?

The precise form of the project is set in consultation with the advisor. However, all students are required to satisfy the following requirements by noon on the day that reading period ends:

- Use the script `/c/cs490/bin/abstract` to submit your name, the title of your project, your advisor's name, and a 250-to-300-word abstract. This information will be added to the on-line database of recent CPSC 490 projects (see `http://zoo.cs.yale.edu/classes/cs490/`).
- Use the script `/c/cs490/bin/submit` to submit a set of web pages describing your project, including a copy of the description submitted with the CPSC 490 form. These pages will become part of the on-line database of recent projects (see `http://zoo.cs.yale.edu/classes/cs490/`).
- Give the SCA a paper copy of your final written report. This document will be added to the (circulating) library of recent CPSC 490 reports.

Note: You must satisfy these requirements even if you plan to continue your project the next term. The only difference is that your electronic abstract, written report, and web pages should constitute an interim progress report (i.e., the level of detail must be the same as in the final versions, but the work described need not be complete).

How do I choose a project?

There are two general approaches

- *student sells project to professor:* you get an idea, write a 3-page prospectus that describes the scope of the project and includes a list of deliverables, and find a faculty member willing to supervise the work (which may require changes in the prospectus)
- *professor sells project to student:* a faculty member has a list of possible projects, and you select one (which may involve changes in the nature of the project)

and a host of possibilities in between.

What kind of project is appropriate?

The project should be more than just an extended homework assignment or final course project and should require that you learn more about some area of computer science. To give you some idea of the range of possibilities, the titles, abstracts, and web pages for recent projects are available on-line (see `http://zoo.cs.yale.edu/classes/cs490/`).

Regular courses meet $2\frac{1}{2}$ hours per week and require 2 to 3 additional hours per week for each hour of class. Using this as a guideline for what it takes to earn a course credit at Yale, the project should be something that you can complete in one semester (i.e., 14 weeks) working approximately 7 to 10 hours per week (i.e., in a total of 100–140 hours).

Note: You cannot be paid for your work on the project. Moreover, to allow others to build on your results, all code and data must be made available to the Yale community.

Who may advise a CPSC 490 project?

The official advisor (and thus the person who evaluates the work and assigns the grade) must be a faculty member with an appointment in the Department of Computer Science. However, the *de facto* advisor need not be, as long as the student meets with the official advisor at least once a month.

Joint majors with Electrical Engineering, Mathematics, and Psychology must have a co-advisor with an appointment in that department.

How can I learn more about projects from past semesters?

The course web pages `http://zoo.cs.yale.edu/classes/cs490/` contain the titles, abstracts, and web pages for recent projects. Copies of the written reports are kept in a circulating library managed by the departmental registrar (AKW 204).

When should I take CPSC 490?

Most students take the course during their final term as the capstone of the program. However, students applying to graduate school should take it in the fall (or, with permission of the DUS, in the spring of their junior year) so that they can get a letter of recommendation from their advisor.

Ideally, planning for the project should begin the preceding term (at least to the extent of finding an advisor).

How often may I take CPSC 490?

While you may not count the course as an elective in any of the computer science majors, you may take it more than once for Yale credit.

May I do a two-term project?

Yes. However, you must satisfy the end-of-term requirements at the end of *each* term, and your grade for each semester will be assigned at the end of *that* semester and will reflect what you accomplished. Thus in effect a two-term project is equivalent to two one-term projects, except that the work may be incomplete at the end of the first semester and the electronic abstract, written report, and web pages for the second semester describe the entire project.

Are group projects allowed?

Yes. However, each member of the group must work on a different part of the project, and *your* description, electronic abstract, final written report, and web pages must focus on *your own* contributions.

What are the "deliverables?"

Whatever you and your advisor decide you must complete by the end of the project. Possibilities include (but are not limited to) code, theorems, simulation studies, data analysis, written reports, and oral presentations.

5.2.2 Projects, Fall 2012

Emma Alexander, *When Shading Flows with Color*, Advisor: Steven Zucker

Caroline Bank, *Gender Effects in Human-Robot Collaboration*, Advisor: Brian Scassellati

Yuval Bussi, *CCD Image Acquisition, Calibration, and Processing for Hyperspectral Applications*, Advisor: Holly Rushmeier

Jacob Evelyn, *A Website for Continuous, Customizable, Dynamically-Generated Music Mashups*, Advisor: Paul Hudak

Peter Lewis, *Diminuendo: Visualizing Self-Similar Music in Haskell*, Advisor: Paul Hudak

Xinyang (Ethan) Li, *Designing a High-Precision Data Collection and Generation System for a Biosensor Device*, Advisors: Mark Reed and Holly Rushmeier

David Meierfrankenfeld, *Attempts at Lower-Stretch Spanning Trees*, Advisor: Daniel Spielman

Rachel Rudinger, *Recognizing Textual Entailment: A Deep Semantic Approach via Dependency Parsing*, Advisors: Dana Angluin and Robert Frank

Samuel Spaulding, *A NLP Package for Semantic Social Referencing*, Advisor: Brian Scassellati

Kartik Venkatraman, *Query Optimization For Distributed Graph Pattern Matching*, Advisor: Daniel Abadi

Christina Wallin, *A Scientific Toolkit for Astronomy Databases*, Advisor: Daniel Abadi

Lewen Yu, *Fold-based Modelling: Investigation into the Origami-inspired*

3D Modelling Paradigm, Advisor: Julie Dorsey

5.2.3 Projects, Spring 2013

Siddhartha Banerjee, *Synchronization and Collective Behaviour*, Advisors: Kumpati Narendra and Stephen Zucker

Michael Brandt, *A Model for Peer-to-Peer Mobile Interactions and Network Bandwidth Reservation*, Advisor: Richard Yang

Nathanael Deraney, *Moonscape: Procedural Planetary Terrain Rendering*, Advisor: Holly Rushmeier

Hari Ganesan, *The Use of Artificial Intelligence in Teaching: Bridge and Other Trick-taking Card Games*, Advisor: Dana Angluin

Bay Gross, *An Extensible Package for the Creation of Generalized "StreamGraph" Visualizations In Ruby*, Advisor: Brad Rosen

Stephen Grugett, *Bayes and Beyond: An Improved Feature-Set for Naive Bayes Classification and a Novel Vector Space Classification Technique for Sentiment Analysis*, Advisor: Drew McDermott

Michael Holkesvik, *The Traveling Spaceman Problem: Moving-Body TSPs and Their Consequences*, Advisor: James Aspnes

Brandon Jackson, *The Codes of Curvature: Reconstructing Macaque V4 Receptive Fields with Gabor Wavelet Noise*, Advisor: Steven Zucker

Caroline Jaffe, *Recognizing and Rewarding Intentionality in Personal Fitness Tracker Usage*, Advisor: Brian Scassellati

Mike Jin, *Fitting Graphs to Vector Data*, Advisor: Daniel Spielman

Kevin Lai, *Implementing Spectral Methods for Multi-way Sparse Cuts*, Advisor: Daniel Spielman

Austin Lan, *A Configuration Based Puzzle Platform Game and Game Level Design Tools,* Advisor: Holly Rushmeier

Daniel Levine, *A Framework for Incentivized Streaming Traffic Optimization*, Advisor: Richard Yang

Michael Levine, *Configuring PostgreSQL for Append-Only File Systems*, Advisor: Daniel Abadi

Danqing Liu, *AnonTalk—an Anonymous Communication Application Powered by Dissent*, Advisor: Bryan Ford

Jacob Metrick, *Building Application-Layer Datagram Protocols on Minion: Disabling TCP Congestion Control and Building uTP on uCOBS on uTCP*, Advisor: Bryan Ford

Daniel Qu, *The Website Genome Project*, Advisor: Richard Yang

Rachel Rudinger, *Recognizing Textual Entailment II: An Extension of a Deep Semantic Approach via Dependency Parsing*, Advisors: Dana Angluin and Robert Frank

Samer Sabri, *Building a Usable Interface and a Computer Opponent for a New Game*, Advisor: James Aspnes

Rebecca Schlussel, *Extended Rounds in the Buddies Anonymous Communication System*, Advisor: Bryan Ford

Benjamin Silver, *Reforming the eBook Peer Lending Regime*, Advisor: Joan Feigenbaum

Samuel Spaulding, *Robotic Inference of Object Sentiment from Natural Language*, Advisor: Brian Scassellati

Sirui Sun, *Developing a Javascript Library for Inter-Language Shared Memory with Deterministic Parallelism*, Advisor: Bryan Ford

Aayush Upadhyay, *Optimistic Snapshot Consistency for Lock-Free Concurrency in Data Structures*, Advisor: Bryan Ford

Christopher Vasseur, *Benefits of Privacy and Anonymity in Information Technology*, Advisor: Joan Feigenbaum

Christina Wallin, *Random Nonce Generation in Dissent*, Advisor: Bryan Ford

Lewen Yu, *On Network Policy Composition and the Maple SDN Controller*, Advisor: Richard Yang

Wayne Zhu, *Keys to Victory: Predicting NFL Outcomes via Machine Learning*, Advisor: Drew McDermott

5.3 Combined B.S. / M.S. in Computer Science

Exceptionally able and well-prepared students may complete a course of study leading to the simultaneous award of the Bachelor of Science and Master of Science degrees after eight terms of enrollment. The requirements are as follows:

- Candidates must satisfy the Yale College requirements for the B.S. degree in Computer Science.
- In fulfilling these requirements, students must complete eight graduate courses from the approved list, up to two of which may, with the permission of the director of undergraduate studies and the director of graduate studies, also be applied toward completion of the B.S. degree. Since the student will be taking CPSC 490a or b *Special Projects*, at most one of

these courses may be CPSC 690a, 691b, or 692a or b *Independent Project.*

Graduate work must not be entirely concentrated in the final two terms of study, and students must take at least six term courses outside computer science during their last four terms at Yale and at least two undergraduate courses during their last two terms.

Students must apply for admission to this program through the director of undergraduate studies no later than the first day of classes of their third-to-last term in Yale College. Applicants must have achieved A, A–, or Honors grades in at least two-thirds of their course credits as well as in at least three-fourths of all course credits that directly relate to computer science.

Interested students are advised to consult with their class advisor, the director of undergraduate studies, and the director of graduate studies by the start of their junior year.

5.4 B.S. in Computer Science and Mathematics

The joint major in Computer Science and Mathematics is intended for students who are interested in computational mathematics, the use of computers in mathematics, mathematical aspects of algorithm design and analysis, and theoretical foundations of computing.[5]

The major requires fourteen term courses as well as a senior project: six required courses (which include the core of the computer science major)

- ◇ CPSC 201a or b *Introduction to Computer Science*
- ◇ CPSC 223b *Data Structures and Programming Techniques*
- ◇ CPSC 323a *Introduction to Systems Programming and Computer Organization*
- ◇ CPSC 365b *Design and Analysis of Algorithms*
- ◇ MATH 120a or b *Calculus of Functions of Several Variables*
- ◇ MATH 244a *Discrete Mathematics;*

a course in linear algebra, one of

- ◇ MATH 222a or b *Linear Algebra with Applications*
- ◇ MATH 225a or b *Linear Algebra and Matrix Theory;*

an advanced course in mathematical computer science, one of

- ◇ CPSC 440b *Numerical Computation*
- ◇ CPSC 455a *Economics and Computation*
- ◇ CPSC 462a *Graphs and Networks*
- ◇ CPSC 465a *Theory of Distributed Systems*
- ◇ CPSC 468a *Computational Complexity*
- ◇ CPSC 469b *Randomized Algorithms*

[5]Students with interests less squarely focused on computer science and extending to applications may prefer the computer science track in the Applied Mathematics major.

Table 3: Sample Computer Science and Mathematics programs.

Fall	*Spring*		*Fall*	*Spring*
CPSC 201 MATH 120	CPSC 223 MATH 225	*Freshman*	CPSC 112 MATH 115	CPSC 201 MATH 120
CPSC 323 MATH 244	CPSC 365 Math elective	*Sophomore*	MATH 222 MATH 244	CPSC 223 Math elective
CS elective Math elective	CS elective Math elective	*Junior*	CPSC 323 Math elective	CPSC 365 Math elective
Math elective Thesis	Math elective	*Senior*	CS elective Math elective	CS elective Math elective Thesis

one additional advanced course in Computer Science; and five additional advanced courses in Mathematics numbered above MATH 200.

Students may substitute MATH 230a and 231b *Vector Calculus and Linear Algebra I and II* for MATH 120a or b and MATH 222a or b or 225a or b. Neither CPSC 480a or b *Directed Reading* nor CPSC 490a or b *Special Projects* nor MATH 470a or b *Individual Studies* may be used as an elective.

The senior requirement is a project or a paper on a topic acceptable to *both* departments. Students must submit a written report (including an electronic abstract and web page(s)) to the Computer Science department, and present an oral report on the mathematical aspects of the project to the Mathematics faculty. If taken for course credit as CPSC 490a or b or MATH 470a or b, the senior project course is in addition to the fourteen required courses.

Table 3 shows typical programs for students who place into MATH 120a or b and have the equivalent of one term of programming experience (left), or who place into MATH 115a or b and have little or no programming experience (right).

The entire program of a student majoring in Computer Science and Mathematics must be approved by the directors of undergraduate studies[6] in *both* departments.

All courses counting toward the major must be taken for a letter grade.

[6]In Computer Science the class advisor acts as the DUS (see §6.1).

SUMMARY OF REQUIREMENTS

Prerequisites: None

Number of courses: Fourteen term courses taken for a letter grade (not including the senior project)

Specific courses required: CPSC 201a or b, 223b, 323a, 365b, one of 440b, 455a, 462a, 465a, 468a, or 469b; MATH 120a or b, either 222a or b or 225a or b, 244a

Distribution of courses: One additional advanced course in computer science; five additional advanced courses in mathematics

Substitution permitted: MATH 230a and MATH 231b for MATH 120a or b and 222a or b or 225a or b

Senior requirement: Senior project or senior essay on topic acceptable to Computer Science and Mathematics departments; written report on project to Computer Science department; oral report on mathematical aspects of project to Mathematics faculty

5.5 B.A. in Computer Science and Psychology

The joint major in Computer Science and Psychology is intended for students interested in integrating work in these two fields.[7] Each area provides tools and theories that can be applied to problems in the other. Examples of this interaction include cognitive science, artificial intelligence, neural models of computation, and biological perception.

The only formal prerequisite for the major is

◇ PSYC 110a or b *Introduction to Psychology,*

from which students who have scored 5 on the Advanced Placement test in Psychology are exempt. Beyond the prerequisite the major requires fourteen term courses as well as a senior project.

Eight of the fourteen courses must be in computer science, including the core of the computer science major:

◇ CPSC 201a or b *Introduction to Computer Science*
◇ CPSC 202a *Mathematical Tools for Computer Science*
◇ CPSC 223b *Data Structures and Programming Techniques*
◇ CPSC 323a *Introduction to Systems Programming and Computer Organization*
◇ CPSC 365b *Design and Analysis of Algorithms;*

and three advanced courses in artificial intelligence. Students may substitute MATH 244a *Discrete Mathematics* for CPSC 202a. Neither CPSC 480a

[7]Students with interests less squarely focused on computer science and psychology and extending to philosophy, linguistics, or neuroscience may prefer the major in Cognitive Science.

or b *Directed Reading* nor CPSC 490a or b *Special Projects* may be used as electives.

The remaining six courses must be in psychology, including

◇ PSYC 200b *Statistics*

at least one course on data collection (PSYC 210–299); at least two courses from the social science point of view:

List A:

◇ PSYC 125a, 126b, 127a, 128b, 131a, 140a, 150b, 231a, 250a, 260b, 330a, 355a, 356b

and at least one course in cognitive psychology or cognitive science listed under Psychology, e.g.,

List C:

◇ PSYC 120a *Brain and Thought: An Introduction to the Human Brain*
◇ PSYC 130a *Introduction to Cognitive Science*
◇ PSYC 131a *Human Emotion*
◇ PSYC 137a *Language and Mind*
◇ PSYC 140a *Developmental Psychology*
◇ PSYC 160b *The Human Brain*
◇ PSYC 181b *Philosophy and the Science of Human Nature*
◇ PSYC 232Lb *Research Methods in Social Decision Making*
◇ PSYC 260b *Research Methods in Behavioral Genetics*
◇ PSYC 304a *The Mental Lives of Babies and Animals*
◇ PSYC 322a *Evolution of Language*
◇ PSYC 327b *Language and Computation*
◇ PSYC 331b *Neurolinguistics*

Neither PSYC 490a or 491b *Directed Reading* nor PSYC 492a or 493b *Directed Research* may be used as electives.

A second course in cognitive psychology or cognitive science may substitute for one of the courses in artificial intelligence. An additional course in psychology may substitute for PSYC 200b if the student has sufficient background in statistics to pass an examination arranged with the instructor.

The senior project must be taken as CPSC 490a or b or PSYC 492a or 493b, depending upon the advisor's department, and must be acceptable to *both* departments. Students must submit a written report (including an electronic abstract and web page(s)) to the Computer Science department.

The entire program of a student majoring in Computer Science and Psychology must be approved by the directors of undergraduate studies[8] in *both* departments.

No courses in Computer Science and at most one course in Psychology

[8]In Computer Science the class advisor acts as the DUS (see §6.1).

may be take on a Credit/D/Fail basis and count for the major.

SUMMARY OF REQUIREMENTS

Prerequisite: PSYC 110a or b
Number of courses: Fourteen term courses beyond prerequisite taken for a letter grade (not including the senior project; one PSYC course may be taken Cr/D/F)
Specific courses required: CPSC 201a or b, 202a, 223b, 323a, 365b, PSYC 200b
Distribution of courses: Eight courses in computer science, with three advanced courses in AI; six courses in psychology, with at least one from PSYC 210–299, at least two from List A, and at least one from List C
Substitution permitted: For CPSC 202a, MATH 244a; for one course in AI, one additional course in cognitive psychology or cognitive science; for PSYC 200b, one additional course in psychology and an examination arranged with the instructor
Senior requirement: CPSC 490a or b or PSYC 492a or 493b with project approved by DUS in each department

5.6 B.S. Electrical Engineering & Computer Science

The joint major in Electrical Engineering and Computer Science is intended for students who want to integrate work in these two fields. It covers discrete and continuous mathematics; algorithm analysis and design; digital and analog circuits; signals and systems; systems programming; and computer engineering. It provides coherence in its core program, but allows flexibility to pursue technical electives.

The prerequisites for the major are:

- ◇ CPSC 112a or b *Introduction to Programming*
- ◇ MATH 112a or b *Calculus of Functions of One Variable I*
- ◇ MATH 115a or b *Calculus of Functions of One Variable II*
- ◇ MATH 120a or b *Calculus of Functions of Several Variables*
- ◇ PHYS 180a and 181b *University Physics.*

Students who must take CPSC 112a or b should do so during the fall of their freshman year to avoid the time conflict between CPSC 112b and PHYS 181b.

Students may substitute Engineering & Applied Science 151a *Multivariable Calculus for Engineers* or MATH 230a *Vector Calculus and Linear Algebra I* for MATH 120a or b; and PHYS 200a and 201b *Fundamentals of Physics* for PHYS 180a and 181b.

Students who must take MATH 112a or b may substitute PHYS 170a and 171b *University Physics for the Life Sciences* for PHYS 180a and 181b. However, since the B.S. programs in Electrical Engineering and in Engineering Sciences (Electrical) do not allow this substitution, students who wish to retain the option of switching to these programs should postpone physics until their sophomore year.

Fifteen term courses are required beyond the prerequisites: ten required courses (which include the core of the computer science major):

◇ CPSC 201a or b *Introduction to Computer Science*
◇ CPSC 202a *Mathematical Tools for Computer Science*
◇ CPSC 223b *Data Structures and Programming Techniques*
◇ CPSC 323a *Introduction to Systems Programming and Computer Organization*
◇ CPSC 365b *Design and Analysis of Algorithms*
◇ EENG 200a *Introduction to Electronics*
◇ EENG 201b *Introduction to Computer Engineering*
◇ EENG 202a *Communications, Computation and Control*
◇ EENG 203b *Circuits and Systems Design*
◇ Either MATH 222a or b *Linear Algebra* or STAT 241a *Probability Theory;*

four advanced electives, two in Computer Science, two in Electrical Engineering; and a senior project.

Students are encouraged to substitute MATH 244a *Discrete Mathematics* for CPSC 202a. Students may substitute MATH 225a or b *Linear Algebra and Matrix Theory* or MATH 231b *Vector Calculus and Linear Algebra II* for MATH 222a or b.

Electives must be either 300- or 400-level courses in the Departments of Computer Science and Electrical Engineering or approved by the directors of undergraduate studies in *both* departments. Cross-listed classes may be counted as being in either department. CPSC 480a or b *Directed Reading* and CPSC 490a or b *Special Projects* may not be used as electives.

The senior project must be taken as CPSC 490a or b or EENG 471a or 472b *Advanced Special Projects*, depending upon the advisor's department, and must be acceptable to *both* departments. Students must submit a written report (including an electronic abstract and web page(s)) to the Department of Computer Science.

A typical program for a student who has taken the equivalent of one year of calculus in high school and has the equivalent of one term of programming experience is shown in Table 4. A typical program for a student who has had only one term of calculus is shown in Table 5.

The entire program of a student majoring in Electrical Engineering and

Table 4: Sample program for a student with some of the prerequisites.

	Fall	*Spring*
Freshman	EENG 200	EENG 201
	PHYS 180	PHYS 181
	ENAS 151	
Sophomore	CPSC 201	CPSC 223
	EENG 202	EENG 203
		MATH 222
Junior	CPSC 202	CPSC 365
	CPSC 323	EE elective
Senior	CS elective	CS elective
	Senior project	EE elective

Students with no or little programming experience should take CPSC 112 in the *fall* of the freshman year, either postponing EENG 200 until the sophomore year or taking MATH 120 in the spring instead of ENAS 151 in the fall.

Table 5: Sample program for a student with only one term of calculus.

	Fall	*Spring*
Freshman	MATH 115	MATH 120
	PHYS 180	PHYS 181
	CPSC 112	EENG 201
Sophomore	CPSC 201	CPSC 223
	EENG 200	EENG 203
	EENG 202	
Junior	CPSC 202	CPSC 365
	CPSC 323	EE elective
	STAT 241	
Senior	CS elective	CS elective
	EE elective	Senior project

Computer Science must be approved by the directors of undergraduate studies[9] in *both* departments.

All courses counting toward the major must be taken for a letter grade.

SUMMARY OF REQUIREMENTS

Prerequisites: CPSC 112a or b; MATH 112a or b, 115a or b, and 120a or b; PHYS 180a, 181b or 200a, 201b

Number of courses: Fifteen term courses beyond the prerequisites taken for a letter grade (including the senior project)

Specific courses required: CPSC 201a or b, 202a, 223b, 323a, and 365b; EENG 200a, 201b, 202a, and 203b; MATH 222a or b or 225a or b or STAT 241a

Distribution of courses: Four additional 300- or 400-level electives, two in computer science, two in electrical engineering

Substitution permitted: MATH 244a for CPSC 202a; advanced courses in other departments, with permission of both departments

Senior requirement: Independent project (CPSC 490a or b or EENG 471a or 472b) acceptable to both departments

6 Miscellany

6.1 Class Advisors and the DUS

We have designated a computer science faculty member to serve as the advisor for all members of your Class. Your class advisor will meet with you at the start of each term to discuss your selection of courses and sign your schedule. Your advisor will also be available throughout the year to answer questions about the major, sign petitions to double-major, and so forth. Moreover, to provide some continuity, you will usually have the same advisor in both your junior and senior years.

The current class advisors are:

Class	Advisor	Office	E-mail
2014	Dana Angluin	AKW 414	`dana.angluin@yale.edu`
2015	Stanley Eisenstat	AKW 208	`stanley.eisenstat@yale.edu`
2016	Stanley Eisenstat	AKW 208	`stanley.eisenstat@yale.edu`

If your class advisor should be unavailable for an extended period of time, then the Director of Undergraduate Studies, Stanley Eisenstat (AKW 208, email: `stanley.eisenstat@yale.edu`) can answer your questions and sign your course schedule.

[9] In Computer Science the class advisor acts as the DUS (see §6.1).

6.2 Life After Yale

Where Do Students Go?

Yale computer science majors are in high demand, both by employers and by graduate and professional schools. For example, last year's seniors went to the following places:

Google	3	Graduate/Professional	5
Microsoft	2	Startups	4
Other software	9	Other	5
Financial Services	7	Unknown	3

Letters of Recommendation

Prospective employers and graduate/professional schools often ask students to submit letters of recommendation from faculty. Instructors in

- project courses (where you work closely with your advisor),
- small courses (where you are more visible),
- advanced courses (where the demands are greater),
- courses taken by graduate students (with whom you can be compared),
- courses related to the area in which you propose to work or study, and
- courses in which you did well

are generally good choices. However, do not be reluctant to ask any instructor for a recommendation.

The best time to request letters is immediately after completing a course, when memories of you and your performance are freshest. Your college dean's office has standardized forms for letter writers to use and return. These letters are kept on file so that you can have copies sent when needed.

Job Search

Many companies interview at Yale for full-time and summer positions in the computing field. Check with Undergraduate Career Services for details, and watch the `cs-majors` mailing list for additional opportunities (see §6.7).

Previous students have recommended the following books to prepare for software engineering interviews:

- John Mongan, Noah Suojanen, and Eric Giguere, *Programming Interviews Exposed: Secrets to Landing Your Next Job*, 2nd edition
- Gayle McDowell, *Cracking the Coding Interview: 150 Programming Questions and Solutions*
- Adnan Aziz and Amit Prakash, *Algorithms for Interviews*.

There are also a multitude of interview web sites and the MIT short course: *Hacking a Google Interview: Mastering Programming Interview Questions* (`http://courses.csail.mit.edu/iap/interview/materials.php`).

Graduate and Professional School

Many computer science majors go to graduate or professional (i.e., law, business, or medical) school, either immediately after graduation or after working for a few years. In either case it is prudent to have letters of recommendation on file and to have taken any entrance examinations (e.g., the GRE, LSAT, GMAT, or MCAT) before leaving Yale.

Ph.D. programs in computer science generally offer research or teaching assistantships that include tuition and a stipend. You can also apply for fellowships from the National Science Foundation and other organizations. In contrast M.S. programs and professional schools typically do not offer any financial support.

Students interested in graduate school are advised to discuss their plans with their class advisor, the director of undergraduate studies, and the director of graduate studies, preferably no later than the spring of their junior year.

6.3 Undergraduate Research

For a general overview of undergraduate research opportunities at Yale, see the YSER (Yale Science and Engineering Research Program) web site, `http://www.yale.edu/yser`.

There is no organized program within Computer Science. However, students wanting to do research (other than that done to satisfy the senior requirement) can

- Take one or more terms of CPSC 290a or b *Directed Research* under the direction of any of our faculty (which earns Yale credit but does not count toward the requirements of the major).
- Work in a research group during the summer or during the academic year. Such paid positions are not common and are arranged directly between a student and a faculty member.

How soon students can begin research depends on their background and area of interest.

6.4 Undergraduate Prize

The department awards a prize to the graduating Senior majoring in computer science who, in the judgment of the Computer Science faculty, ranks highest in scholarship.

The ranking is based on grade point average in courses that count toward the major and were taken at Yale. To be eligible, a student must have taken at least 12 such courses. The GPA is computed using the formula on which General Honors are based, but using the list of courses that determines eligibility for Distinction in the Major. Thus 100-level courses; 200-level courses other than CPSC 201a or b, 202a, and 223b; CPSC 480a or b; all but the first term of CPSC 490a or b, and courses taken only for the M.S. part of the B.S. / M.S. program are excluded.

The recent winners were

2013	Wayne Zhu
2012	Cameron Musco
2011	Jonathan Eng and Andrew Gu (cowinners)
2010	Michael Mester
2009	Justin Thaler
2008	Andrew Smith

6.5 DSAC

The Departmental Student Advisory Committee (DSAC) represents undergraduate computer science majors and provides a liaison between the faculty and undergraduates in matters pertaining to the computer science curriculum and the majors' use of departmental resources. DSAC also helps with the planning and operation of the Zoo, the undergraduate computing facility for the department (see §4).

As defined by Yale College, DSAC's charter is to review aspects of the department's undergraduate curriculum as it affects both majors and non-majors, and to serve as a channel through which solicited or unsolicited opinions of other students can be expressed. It advises the department on such matters as ideas for new courses and programs; the effectiveness of the curriculum; the scope and sequence of course offerings; the requirements for the major; the role of the senior project; proposals for the improvement of instruction and advising; and the usefulness and interest of specific courses to non-majors.

Feel free to contact DSAC (`dsac@cs.yale.edu`) if you have suggestions about the curriculum, want help using the Zoo, or have general questions about the Department. DSAC also plans several pizza parties throughout the year for majors and other students interested in computer science.

The members of DSAC for 2013–2014 and their e-mail addresses are

Feridun Celebi	`feridun.celebi@yale.edu`
Michael Hopkins	`michael.hopkins@yale.edu`
Apurv Suman	`apurv.suman@yale.edu`
Jessica Tordoff	`jessica.tordoff@yale.edu`
Cyril Zhang	`cyril.zhang@yale.edu`

They are also reachable at `dsac@cs.yale.edu`. The DSAC web page is `http://zoo.cs.yale.edu/dsac`.

6.6 Advice from Graduating Seniors

The Classes of 2013, 2012, and 2011 were asked to pass along some of the insights they gained during their years as a major. Here is an edited selecton:

- Start assignments early and work on them often. They will take less time and they won't take over your life (usually).
- Coding is fun when you have the time, and terrible when you're at deadline.
- Go to office hours. Ask questions. It saves you so much time.
- Work in the Zoo! You will be more productive and can ask and answer questions, order food, play Rock Band, and pull an all-nighter without feeling tired at all.
- Don't be afraid to approach professors or TA's for help on problem sets, difficult topics in class, etc.
- Ask questions in class, even if they seem to be "dumb" questions.
- Strive for learning, not for finishing assignments as fast as possible or just getting good grades.
- Take a wide variety of electives—you never know where you might find your true passion.
- Make sure that your senior project is well defined at the beginning of the semester.
- Try a research project.
- Challenge yourself in the courses that you take.
- Mentoring is a valued skill, wherever you end up. Help other students in the Zoo or apply to be a peer tutor.
- Work on projects with people outside of class.
- Build and launch stuff on the side; learn new languages and explore new technologies.
- If you want a summer internship, start looking in September.
- Apply for internships in the same semester that you take CPSC 223.

- Know algorithms and data structures well if you want to get a good job.
- Get involved in research early if you think it's something that you might be interested in.
- Look outside the department to supplement your CS curriculum.
- Enjoy the wealth of opportunities Yale presents to you and do not limit yourself during your time here as an undergraduate.
- Take a balanced course load, each semester. Dedicate enough time to each of your CS courses, but also explore everything that Yale has to offer.
- Focus on doing what you love.
- Don't schedule every minute of every day. Leave some time each week for spontaneity.
- You will do better and more efficient work after sleeping for a few hours. Don't try to stay up all night to get something done. Take care of yourself!

See `http://dus.cs.yale.edu/blurbs.html` for the full responses.

6.7 Additional Sources of Information

The `cs-majors` mailing list contains postings of interest to undergraduates majoring in computer science or taking courses in the subject, including announcements of new courses and faculty, colloquia and other departmental events, recruiting visits and employment/internship opportunities, as well as messages from the director of undergraduate studies. The mailing list can be accessed on the web via

`http://mailman.cs.yale.edu/mailman/listinfo/cs-majors`

It contains all the archived posts as well as instructions on how to subscribe to or un-subscribe from the mailing list.

7 Course Listings

7.1 Introductory Courses

CPSC 079b *Digital Photorealism.*
(Not taught in 2013–2014)
Examination of basic methods used to define shapes, materials, and lighting when creating computer-generated images. Topics include mathematical models for shape, texture models, and lighting techniques. Principles are applied through use of modeling/rendering software. The term project will be the production of a short animated video with rich visual effects. *Proficiency in high-school–level mathematics is assumed. No previous experience with computers necessary.* (Preregistration required; open only to freshmen.)

CPSC 101b *Great Ideas of Computer Science.* Dana Angluin
TTh 1:00–2:15
An introduction for nonmajors to some of the most important ideas in computer science: What the computer is; how it works; what it can do and what it cannot do, now and in the future. Topics include algorithms, elementary programming, hardware, language interpretation, software engineering, complexity, models of computation, and artificial intelligence. *No previous programming experience required.*

CPSC 112a or b *Introduction to Programming.*

CPSC 112a.	Drew McDermott	MWF 10:30–11:20
CPSC 112b.	Richard Yang	MWF 11:35–12:25

An introductory course designed to teach students majoring in any subject how to program computers. The language taught is either JAVA or C#. The focus is on the development of programming skills, problem-solving methods, and selected applications. Topics include data types, control structures, basic algorithms, object-oriented programming, graphical user interfaces, and some advanced programming concepts. *No previous experience with computers necessary.*

CPSC 150a / HUMS 407a *Computer Science and the Modern Intellectual Agenda.* David Gelernter
MW 11:35–12:50
An introduction to the basic ideas of computer science (computability, algorithm, virtual machine, symbol processing system) and of several ongoing relationships between computer science and other fields, including philosophy of mind, classical cognitivism, connectionism, and artificial life. *No previous experience with computers necessary.* (Satisfies the WR and HU

requirements. Enrollment limited to 25.)

CPSC 151b / HUMS 408b *The Graphical User Interface: DOS to Windows to What?.* David Gelernter
MW 11:35–12:50
The role of Graphical User Interfaces (such as the Desktop, with its overlapping windows, icons, menus and pointer device–as embodied in Mac OS, Microsoft Windows etc), on standard platforms such as desktop PCs, laptops, small-screen devices etc. Why did GUIs develop in the way they did? Why have they evolved so little since the Desktop of the 1970s? How will changing hardware and user requirements reshape them in the future? *Prerequisite: Have used a desktop or laptop computer.* (Satisfies the WR requirement. Enrollment limited to 25.)

CPSC 183a *Introduction to Law, Technology, and Culture.* Brad Rosen
MW 4:00–5:15
An exploration of the myriad of ways in which law and technology intersect, with a special focus on the role of cyberspace. The course lays out a basic framework for the many issues that arise in our modern legal and technological contexts. It covers topics such as digital copyright, free speech, privacy and anonymity, information security, innovation, online communities, the impact of technology on society, and emerging trends. *No technical knowledge or previous coursework required.* (Satisfies the SO requirement.)

CPSC 185b *Control, Privacy, and Technology.* Brad Rosen
F 3:30–5:20
A case-law and policy intensive class that explores how various legal doctrines have evolved with and around technological development. Topics include criminal law, privacy, search and seizure, digital rights, and the implications of technologically-permitted (and enforced) methods of control on the law. *After CPSC* 180a *or* b *or* 183a. (Satisfies the WR and SO requirements. Enrollment limited to 24.)

CPSC 201a or b *Introduction to Computer Science.*

CPSC 201a.	Dana Angluin	MWF 10:30–11:20
CPSC 201b.	Holly Rushmeier	MWF 11:35–12:25

An introduction to the concepts, techniques, and applications of computer science for potential majors. Topics include computer systems (the design of computers and their languages); theoretical foundations of computing (computability, complexity, algorithm design); and artificial intelligence (the organization of knowledge and its representation for efficient search). Examples stress the importance of different problem-solving methods. *After*

CPSC 112a *or* b *or equivalent.*

CPSC 202a *Mathematical Tools for Computer Science.* James Aspnes
TTh 1:00–2:15
Introduction to formal methods for reasoning and to mathematical techniques basic to computer science. Topics include propositional logic, discrete mathematics, and linear algebra. Emphasis on applications to computer science: recurrences, sorting, graph traversal, Gaussian elimination.

CPSC 223b *Data Structures and Programming Techniques.*
Stanley Eisenstat
MW 1:00–2:15
Topics include programming in C; data structures (arrays, stacks, queues, lists, trees, heaps, graphs); sorting and searching; storage allocation and management; data abstraction; programming style; testing and debugging; writing efficient programs. *After CPSC* 201a *or* b *or equivalent.*

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

MATH 244a / AMTH 244a *Discrete Mathematics.* Staff
MW 2:30–3:45
Basic concepts and results in discrete mathematics: graphs, trees, connectivity, Ramsey theorem, enumeration, binomial coefficients, Stirling numbers. Properties of finite set systems. *After MATH* 115a *or* b *or equivalent.*

7.2 Intermediate Courses

CPSC 323a *Introduction to Systems Programming and Computer Organization.* Stanley Eisenstat
MW 1:00–2:15
Machine architecture and computer organization, systems programming in a high-level language, issues in operating systems, software engineering, prototyping in scripting languages. *After CPSC* 223b.

CPSC 365b *Design and Analysis of Algorithms.* Daniel Spielman
TTh 2:30–3:45
Paradigms for problem solving: divide and conquer, recursion, greedy algorithms, dynamic programming, randomized and probabilistic algorithms. Techniques for analyzing the efficiency of algorithms and designing efficient

algorithms and data structures. Algorithms for graph theoretic problems, network flows, and numerical linear algebra. Provides algorithmic background essential to further study of computer science. *After CPSC* 202a *and* 223b.

7.3 Advanced Courses

CPSC 421b *Compilers and Interpreters.* Zhong Shao
TTh 1:00–2:15
Compiler organization and implementation: lexical analysis, formal syntax specification, parsing techniques, execution environment, storage management, code generation and optimization, procedure linkage, and address binding. The effect of language-design decisions on compiler construction. *After CPSC* 323a.

CPSC 422b *Operating Systems.* Bryan Ford
MW 1:00–2:15
The design and implementation of operating systems. Topics include synchronization, deadlock, process management, storage management, file systems, security, protection, and networking. *After CPSC* 323a.

CPSC 423a *Principles of Operating Systems.* Avi Silberschatz
TTh 2:30–3:45
A grand tour of the underlying principles of modern operating systems. Main topics to be covered: Process management, memory management, storage management, protection and security, distributed systems, and virtual machines. The main emphasis is on the presentation of fundamental concepts rather than implementation. *After CPSC* 323a.

CPSC 424b *Parallel Programming Techniques.*
(Not taught in 2013–2014)
Practical introduction to parallel programming, emphasizing techniques and algorithms suitable for scientific and engineering computations. Aspects of processor and machine architecture. Parallel programming techniques including multithreading, message passing, and data parallel computing using graphics processing units (GPUs). Performance measurement, tuning, and debugging of parallel programs. Parallel file systems and I/O. *After CPSC* 223b *and MATH* 222a *or* b *or* 225a *or* b*, or equivalent.*

CPSC 426a *Building Decentralized Systems.* Brian Ford
MW 2:30–3:45
An exploration of the challenges and techniques for building decentralized computing systems, in which many networked computers need to cooperate reliably despite failures and without assuming centralized management. Topics include: decentralized storage systems, mobile and remote execution, hosting untrusted code, [byzantine] fault tolerance, naming, capabilities, information flow control, distributed shared memory, distributed hash tables, content distribution, practical uses of cryptography. This course is programming-intensive. *After CPSC* 323a.

CPSC 427a *Object-oriented Programming.*
(Not taught in 2013–2014)
Object-oriented programming as a means to efficient, reliable, modular, reusable code. Use of classes, derivation, templates, name-hiding, exceptions, polymorphic functions, and other features of C++. *After CPSC* 223b.

CPSC 428a *Language-Based Security.*
(Not taught in 2013–2014)
Basic design and implementation of language-based approaches for increasing the security and reliability of systems software. Topics include proof-carrying code; certifying compilation; typed assembly languages; runtime checking and monitoring; high-confidence embedded systems and drivers; and language support for verification of safety and liveness properties. *After CPSC* 202a *and* 323a, *or equivalent.*

CPSC 430a *Formal Semantics.* Zhong Shao
MW 11:35–12:50
Introduction to formal approaches to programming language design and implementation. Topics include the lambda-calculus, type theory, denotational semantics, type-directed compilation, higher-order modules, and application of formal methods to systems software and Internet programming. *After CPSC* 202a *and* 323a. (Not taught every year.)

CPSC 431b *Computer Music—Algorithmic and Heuristic Composition.*
(Not taught in 2013–2014)
Study of the theoretical and practical fundamentals of computer-generated music, with a focus on high-level representations of music, algorithmic and heuristic composition, and programming languages for computer music generation. Theoretical concepts are supplemented with pragmatic issues expressed in a high-level programming language. *After CPSC* 202a *and* 223b. *Assumes ability to read music.* (Taught in alternate years.)

CPSC 432b *Computer Music—Sound Representation and Synthesis.*
Paul Hudak
MW 2:30–3:45
Beginning with low-level representations of sound, various methods for synthesizing musical sounds are studied, including additive synthesis, subtractive synthesis, frequency modulation, granular synthesis, and physical modeling. The goal is to both simulate as accurately as possible existing musical instruments, and to create new sounds and musical soundscapes. Scales and tuning systems are also studied, as is basic acoustic signal processing (filtering, reverb, sound effects, etc). A key idea underlying the course is the use of a high-level functional language to express the theoretical concepts in a practical manner. Regular programming assignments lead toward a final project that is a software realization of a student-designed concept. *After CPSC* 202a *and* 223b. *Assumes ability to read music.* (Taught in alternate years.)

CPSC 433a *Computer Networks.* Yang Richard Yang
TTh 1:00–2:15
An introduction to the design, implementation, analysis, and evaluation of computer networks and their protocols. Topics include layered network architectures, applications, transport, congestion, routing, data link protocols, local area networks, performance analysis, multimedia networking, network security, and network management. Emphasis on protocols used in the Internet. *After CPSC* 323a. (Taught in alternate years.)

CPSC 434a *Mobile Computing and Wireless Networking.*
(Not taught in 2013–2014)
An introduction to the principles of mobile computing and its enabling technologies. Topics include principles of mobile computing; wireless systems; information management; location-independent/dependent computing models; disconnected and weakly connected operation models; human-computer interactions; mobile applications and services; security; power management; and sensor networks. *After CPSC* 202a *and* 323a. (Taught in alternate years.)

CPSC 437b *Introduction to Database Systems.* Avi Silberschatz
TTh 2:30–3:45
An introduction to database systems. Data modeling. The relational model and the SQL query language. Relational database design, integrity constraints, functional dependencies, and normal forms. Object-oriented databases. Database data structures: files, B-trees, hash indexes. *After CPSC* 202a *and* 223b.

CPSC 438b *Database System Implementation and Architectures.*
(Not taught in 2013–2014)
A study of systems programming techniques, with a focus on database systems. Half the course is spent studying the design of a traditional DBMS, supplemented by a hands-on exercise where students build various components (e.g., a catalog-manager, a buffer-manager, and a query execution engine) of a DBMS prototype. The other half is spent on non-traditional architectures (parallel databases, data warehouses, stream databases, Web databases). *After CPSC* 202a *and* 323b.

CPSC 440b *Numerical Computation.* Vladimir Rokhlin
TTh 1:00–2:15
Algorithms for numerical problems in the physical, biological, and social sciences: solution of linear and nonlinear systems of equations, interpolation and approximation of functions, numerical differentiation and integration, optimization. *After CPSC* 112a *or* b *or an equivalent introductory programming course; MATH* 120a *or* b*; and MATH* 222a *or* b *or* 225a *or* b *or CPSC* 202a.

CPSC 445a *Introduction to Data Mining.* Vladimir Rokhlin
MW 1:00–2:15
A study of algorithms and systems that allow computers to find patterns and regularities in databases, to perform prediction and forecasting, and to improve their performance generally through interaction with data. *After CPSC* 202a *and* 223b *and MATH* 222a *or* b, *or equivalents.*

CPSC 455a / ECON 425a *Economics and Computation.*
(Not taught in 2013–2014)
A mathematically rigorous investigation of the interplay of economic theory and computer science with an emphasis on the relationship of incentive-compatibility and algorithmic efficiency. Particular attention will be paid to the formulation and solution of mechanism-design problems that are relevant to data networking and Internet-based commerce. *Familiarity with basic microeconomic theory is helpful but not required. After CPSC* 365b *or with permission of the instructor.*

CPSC 457a *Sensitive Information in a Wired World.* Joan Feigenbaum
TTh 1:00–2:15
An examination of issues of ownership, control, privacy, and accuracy of the huge amount of sensitive information about people and organizations that is collected, stored, and used by today's ubiquitous information systems. Readings consist of research papers that explore both the power and the

limitations of existing privacy-enhancing technologies such as encryption and “trusted platforms.” *Recommended to be taken after or concurrently with CPSC* 365b *and* 467b. (Not taught every year.)

CPSC 462a / AMTH 462a *Graphs and Networks.* Daniel Spielman
TTh 2:30–3:45
A mathematical examination of graphs and their applications in the sciences. Families of graphs include social networks, small-world graphs, Internet graphs, planar graphs, well-shaped meshes, power-law graphs, and classic random graphs. Phenomena include connectivity, clustering, communication, ranking, and iterative processes. *Prerequisites: Linear algebra and discrete mathematics; a course in probability is recommended.* (Not taught every year.)

CPSC 465b *Theory of Distributed Systems.* James Aspnes
MWF 11:35–12:25
Models of asynchronous distributed computing systems. Fundamental concepts of concurrency and synchronization, communication, reliability, topological and geometric constraints, time and space complexity, and distributed algorithms. *After CPSC* 323a *and* 365b. (Taught in alternate years. Formerly CPSC 425b.)

CPSC 467a *Cryptography and Computer Security.* Michael Fischer
MW 2:30–3:45
A survey of such private and public key cryptographic techniques as DES, RSA, and zero-knowledge proofs, and their application to problems of maintaining privacy and security in computer networks. Focus on technology, with consideration of such societal issues as balancing individual privacy concerns against the needs of law enforcement, vulnerability of societal institutions to electronic attack, export regulations and international competitiveness, and development of secure information systems. *Some programming may be required. After CPSC* 202a *and* 223b.

CPSC 468a *Computational Complexity.*
(Not taught in 2013–2014)
Introduction to the theory of computational complexity. Basic complexity classes, including Polynomial Time, Nondeterministic Polynomial Time, Probabilistic Polynomial Time, Polynomial Space, Logarithmic Space, and Nondeterminstic Logarithmic Space. The roles of reductions, completeness, randomness, and interaction in the formal study of computation. *After CPSC* 365b *or with permission of the instructor.*

CPSC 469b *Randomized Algorithms.*
(Not taught in 2013–2014)
Beginning with an introduction to tools from probability theory including some inequalities like Chernoff bounds, the course will cover randomized algorithms from several areas: graph algorithms, algorithms in algebra, approximate counting, probabilistically checkable proofs, and matrix algorithms. *After CPSC* 365b*; a solid background in probability is desirable.* (Taught in alternate years.)

CPSC 470a *Artificial Intelligence.*
(Not taught in 2013–2014)
An introduction to artificial intelligence research, focusing on reasoning and perception. Topics include knowledge representation, predicate calculus, temporal reasoning, vision, robotics, planning, and learning. *After CPSC* 201a *or* b *and* 202a.

CPSC 471b *Topics in Artificial Intelligence.* Drew McDermott
MW 2:30–3:45
An in-depth study of one area of artificial intelligence. Possible topics include automated planning, scheduling with explicitly represented objective functions, AI and philosophy of mind, computational approaches to stereo vision, multiagent systems, and automated diagnosis using functional models. *After CPSC* 470a *or with permission of instructor.* (Not taught every year.)

CPSC 472a *Intelligent Robotics.* Brian Scassellati
MWF 10:30–11:20
An introduction to the construction of intelligent, autonomous systems. Sensory-motor coordination and task-based perception. Implementation techniques for behavior selection and arbitration, including behavior-based design, evolutionary design, dynamical systems, and hybrid deliberative-reactive systems. Situated learning and adaptive behavior. *After or concurrently with CPSC* 201a *or* b *and* 202a. (May not be taken after CPSC 473b)

CPSC 473b *Laboratory in Intelligent Robotics.* Brian Scassellati
MWF 10:30–11:20
A laboratory experience in intelligent robotics. Students will work in small teams to construct novel research projects using one of a variety of robot architectures. Project topics will be developed during the first weeks of the course and may include topics in human-robot interaction, adaptive intelligent behavior, active perception, humanoid robotics, and socially assistive robotics. *After CPSC* 472a. *CPSC* 223b *is recommended.*

CPSC 475a / BENG 475a *Computational Vision and Biological Perception.* Steven Zucker
MW 2:30–3:45
An overview of computational vision with a biological emphasis. Suitable as an introduction to biological perception for computer science and engineering students, as well as an introduction to computational vision for mathematics, psychology, and physiology students. *After CPSC* 112a *or* b *and MATH* 120a *or* b, *or with permission of the instructor.* (Satisfies the SC requirement.)

CPSC 478b *Computer Graphics.*
(Not taught in 2013–2014)
An introduction to the basic concepts of two- and three-dimensional computer graphics. Topics include affine and projective transformations, clipping and windowing, visual perception, scene modeling and animation, algorithms for visible surface determination, reflection models, illumination algorithms, and color theory. Assumes solid C or C++ programming skills and a basic knowledge of calculus and linear algebra. *After CPSC* 202a *and* 223b.

CPSC 479b *Advanced Topics in Computer Graphics.* Julie Dorsey
MW 1:00–2:15
An in-depth study of advanced algorithms and systems for rendering, modeling, and animation in computer graphics. Topics vary and may include reflectance modeling, global illumination, subdivision surfaces, NURBS, physically-based fluids systems, and character animation. *After CPSC* 202a *and* 223b. (Not taught every year.)

CPSC 480a or b *Directed Reading.* By arrangement with faculty
Individual study for qualified students who wish to investigate an area of computer science not covered in regular courses. A student must be sponsored by a faculty member who sets the requirements and meets regularly with the student. Requires a written plan of study approved by the faculty advisor and the director of undergraduate studies. *May be taken more than once for credit.*

CPSC 490a or b *Special Projects.* By arrangement with faculty
Individual research. Requires a faculty supervisor and the permission of the class advisor or the director of undergraduate studies. The student must submit a written report about the results of the project. *May be taken more than once for credit.*

7.4 Graduate Courses

Graduate courses and seminars may be open to undergraduates. Check with the instructor and with your class advisor or the DUS.

7.5 Related Courses in Other Departments

EENG 201b *Introduction to Computer Engineering.*

EENG 348a *Digital Systems.*

EENG 425a *Introduction to VLSI System Design.*

LING 227b *Language and Computation.*

MATH 222a or b / AMTH 222a or b *Linear Algebra with Applications.*

MATH 225a or b *Linear Algebra and Matrix Theory.*

MATH 270a *Set Theory.*

MATH 345b *Modern Combinatorics.*

OPRS 235a / AMTH 235a *Optimization.*

PHIL 267a *Mathematical Logic.*

PHIL 427a *Computability and Logic.*

STAT 230b *Introductory Data Analysis.*

STAT 241a / MATH 241a *Probability Theory.*

STAT 242b / MATH 242b *Theory of Statistics.*

STAT 251b / MATH 251b / ENAS 496b *Stochastic Processes.*

STAT 364b / AMTH 364b / EENG 454b *Information Theory.*

STAT 365b *Data Mining and Machine Learning.*

8 Course Planner

Course Requirements

B.S./B.A. in Computer Science:
CPSC 201a or b, 202a, 223b, 323a, 365b
6 CPSC electives (B.S.) / 4 CPSC electives (B.A.); senior thesis

B.S. in Computer Science and Mathematics
CPSC 201a or b, 223b, 323a, 365b; 2 CPSC electives;
MATH 120, 222/225, 244; 5 MATH electives; senior thesis

B.A. in Computer Science and Psychology
Prerequisites: PSYC 110a or b
CPSC 201a or b, 202a, 223b, 323a, 365b; 3 AI electives
PSYC 200b, 210–299; 4 PSYC electives; senior thesis

B.S. in Electrical Engineering and Computer Science
Prerequisites: CPSC 112a or b; MATH 120a or b; PHYS 180a, 181b
CPSC 201a or b, 202a, 223b, 323a, 365b; 2 CPSC electives
EENG 200a, 201b, 202a, 203b; 2 EENG electives
MATH 222a or b / 225a or b / 241a; senior thesis

Fall		**Spring**
	Freshman	
	Sophomore	
	Junior	
	Senior	

9 Yale College Calendar / Deadlines

2013		**Fall Term**
28 Aug.	Wed.	Fall-term classes begin
30 Aug.	Fri.	Friday classes do not meet; Monday classes do meet
2 Sep.	Mon.	Labor Day; classes do not meet
9 Sep.	Mon.	Course schedules due for freshmen
10 Sep.	Tue.	Course schedules due for sophomores/juniors
11 Sep.	Wed.	Course schedules due for seniors
19 Sep.	Thu.	CPSC 490 forms due, Noon
18 Oct.	Fri.	Midterm; last day to withdraw from a course without having the course appear on the transcript
22 Oct.	Tue.	October recess begins, 11:00 PM
28 Oct.	Mon.	Classes resume
8 Nov.	Fri.	Last day to convert from CR/D/F to a letter grade
22 Nov.	Fri.	Fall recess begins, 5:30 PM
2 Dec.	Mon.	Classes resume
6 Dec.	Fri.	Classes end, 5:30 PM; reading period begins
		Last day to withdraw from a fall-term course
11 Dec.	Wed.	CPSC 490 projects due, Noon
12 Dec.	Thu.	Final examinations begin, 9:00 AM
17 Dec.	Tue.	Examinations end, 5:30 PM; winter recess begins

2014		**Spring Term**
13 Jan.	Mon.	Spring-term classes begin
17 Jan.	Fri.	Friday classes do not meet; Monday classes do meet
20 Jan.	Mon.	Martin Luther King Jr. Day; classes do not meet
22 Jan.	Wed.	Course schedules due for freshmen
23 Jan.	Thu.	Course schedules due for sophomores/juniors
24 Jan.	Fri.	Course schedules due for seniors
6 Feb.	Thu.	CPSC 490 forms due, Noon
7 Mar.	Fri.	Midterm; last day to withdraw from a course without having the course appear on the transcript
		Spring-term recess begins, 5:30 PM
24 Mar.	Mon.	Classes resume
4 Apr.	Fri.	Last day to convert from CR/D/F to a letter grade
25 Apr.	Fri.	Classes end, 5:30 PM; reading period begins
		Last day to withdraw from a spring-term course
30 Apr.	Wed.	CPSC 490 projects due, Noon
1 May	Thu.	Final examinations begin, 9:00 AM
6 May	Tue.	Examinations end, 5:30 PM
19 May	Mon.	University Commencement

Inquiries concerning the contents of this handbook may be referred to:

DIRECTOR OF UNDERGRADUATE STUDIES
DEPARTMENT OF COMPUTER SCIENCE
YALE UNIVERSITY
P.O. BOX 208285
NEW HAVEN, CT 06520–8285

Email: `ugradinfo@cs.yale.edu`

Phone: 203–432–1283

Web: `http://ugrad.cs.yale.edu`

Last revised July 2013.