COMPUTER SCIENCE

haverford.edu/computer-science

Computer science is the representation and manipulation of information; it is the study of the theory, analysis, design, and implementation of the data structures that represent information and the algorithms that transform them. Computer science is interdisciplinary, with roots in mathematics, physics, and engineering, and with applications in virtually every academic discipline and professional enterprise.

Computer science at Haverford College covers these fundamental concepts, with emphasis on depth of thought, clarity of expression and attention to ethical impact. This approach is consistent with the principles of scientific education in the liberal arts. Our aim is to provide students with a base of skills and capabilities that support a wide variety of post-graduation goals, rather than to follow short-term fashions and fluctuations in computer hardware and software.

LEARNING GOALS

Each student in computer science will be able to:

- **Realize their full ability to think deeply.** This involves mastering discipline-specific concepts such as abstraction, correctness, and complexity, and recognizing their broad and deep applications, both theoretically and practically, in new contexts.
    - Identify the role of abstraction in a computational problem situation; for example, distinguish a general problem from a specific problem instance, or understand the mapping between an abstract data type (ADT) and a given representation of that ADT.
    - Develop original, correct solutions demonstrating an appropriate level of abstraction, using two or more design techniques specific to the field.
    - Express a general solution in an appropriate programming language.
    - Analyze and compare the efficiency of alternative solutions, both quantitatively and qualitatively.
    - Increase confidence in a solution through a variety of approaches, including code review, testing, and mathematical reasoning.
- **Communicate their thinking clearly and effectively.** This involves taking a discovered or developed solution (or a given problem definition, etc.) and sharing that solution with peers, managers, clients, and other professionals, in a complete and persuasive manner, and with appropriate use of vocabulary and other tools (e.g., charts, proofs, demonstrations).
- **Identify, interpret and evaluate the theoretical, practical, and ethical implications of their work in the field.** This work is most easily identified as software, but other results might be papers written and published, projects chosen over others ignored, and even questions raised during the software development process.

CURRICULUM

Computer science offers:

- a major.
- a concentration for mathematics majors.
- a minor.

Computer science also contributes substantially to the Concentration in Scientific Computing. More information on this concentration can be found on the program's website (*haverford.edu/scientific-computing*) or catalog entry.

The major in computer science is designed for students who wish to explore fundamental questions about computation and the role of computation in society. As part of this exploration, we provide many opportunities for students to design, implement, and analyze algorithms and data structures, and develop a larger-scale hardware/software system over the course of multiple semesters. These opportunities include both individual projects and group work, and provide experience with a variety of programming languages and with computer hardware. The senior experience, and the final projects in many classes, provide opportunities for students to explore their own interests in computer science.

MAJOR REQUIREMENTS

The major program covers the foundations of the discipline and provides a range of elective opportunities. While the computer science major

is inspired by guidance from existing professional societies in computing, it is uniquely "Haverfordian" in its emphasis on a collaborative approach to a rigorous field of inquiry.

Requirements are:

- CMSC 105 (Introduction to Computer Science) *or* CMSC 107 *or* Bryn Mawr CMSC 110.
- CMSC 106 (Introduction to Data Structures) *or* CMSC 107 *or* Bryn Mawr CMSC 206.
- CMSC/MATH 231 (Discrete Mathematics) (Students with strong backgrounds in mathematics and prior knowledge of the topics covered in Math/CMSC 231 may wish to seek instructor permission to place into CMSC 340 / 345 without prior completion of 231 — in this case, the student may complete the requirements for the major with another course covering discrete mathematics, from the following list: Linear Optimization (MATH 210b), Logic (MATH 394), Cryptography (MATH 394), Combinatorics (MATH 395), or STAT 203, 218, 286, or 396).
- CMSC 240 (Principles of Computer Organization).
- CMSC 245 (Principles of Programming Languages).
- CMSC 340 (Analysis of Algorithms).
- CMSC 345 (Theory of Computation)
- CMSC 350 (Compiler Design) *or* CMSC 355 (Operating Systems) *or* CMSC 356 (Concurrency and Co-Design in Operating Systems).
- One additional 300 level course in computer science, and two additional courses numbered 200 or higher (one may be a related course in math or physics); computer science independent study courses can be counted if the student has received permission in advance to do so.
- CMSC 399 Senior Thesis and Seminar.

MINOR REQUIREMENTS

- CMSC 105 (Introduction to Computer Science) *or* CMSC 107 *or* Bryn Mawr CMSC 110.
- CMSC 106 (Introduction to Data Structures) *or* CMSC 107 *or* Bryn Mawr CMSC 206.
- CMSC/MATH 231 (Discrete Mathematics) (Students with strong backgrounds in mathematics and prior knowledge of the topics covered in CMSC/MATH 231 may wish to seek instructor permission to place into CMSC 340 / 345 without prior completion of 231—in this case, the student may complete the requirements for the minor with another course covering discrete mathematics, from the following list: MATH 210b (Linear Optimization), MATH 394 (Logic), MATH 394 (Cryptography), MATH 395 (Combinatorics), or STAT 203, 218, 286, or 396).
- *Either*
  - CMSC 240 (Principles of Computer Organization) *and* a course on operating systems [i.e., *either* CMSC 355 (Operating Systems) *or* CMSC 356 (Concurrency and Co-Design in Operating Systems)], *or*
  - CMSC 245 (Principles of Programming Languages) *and* CMSC 350 (Compiler Design).
- *Either* CMSC 340 (Analysis of Algorithms) *or* CMSC 345 (Theory of Computation).

CONCENTRATION REQUIREMENTS

The Computer Science Department supports the Concentration in Scientific Computing, available to a variety of majors (*haverford.edu/scientific-computing*), and provides a computer science concentration specific to mathematics majors.

Computer Science Concentration for Mathematics Majors Requirements

- CMSC 105 (Introduction to Computer Science) *and* 106 (Introduction to Data Structures), or CMSC 107.
- *Either* CMSC 240 (Principles of Computer Organization) *or* 245 (Principles of Programming Languages).
- *Either* CMSC 340 (Analysis of Algorithms) *or* 345 (Theory of Computation).
- One cross-listed MATH/CMSC course (Note that MATH/CMSC 231 meets this requirement and is the prerequisite for CMSC 340 and 345.)
- One additional 300-level computer science course.

SENIOR THESIS

The senior thesis in computer science is a capstone experience under the guidance of a faculty member. Students complete a thorough literature review in the initial term, and can continue with a research project into the subsequent term. Oral, poster and written presentations are required. This experience can include original work, but it must demonstrate

deep thinking and an original exposition of an advanced topic.

Students are required to enroll in a one-credit senior seminar course in the Fall term to ensure that they successfully complete this graduation requirement. There is a series of class activities and deadlines to help keep students on track for completing their thesis. In the fall semester, these include: the advisor selection process; submitting the topic proposal; completing the literature review; and the public poster presentation. In the optional spring semester, these include: implementing their project proposed in the previous term; completing a rough draft of their thesis; rehearsing their oral presentations; submitting the final thesis document; and giving their oral presentation. A second reader provides feedback periodically to the student and their advisor as to whether progress is satisfactory.

A detailed schedule is provided to all students in the seminar at the beginning of the year.

Senior Project Learning Goals
The thesis work culminates in the writing and oral presentation of a paper. The student must also demonstrate the research skills required to produce this paper, in accordance with departmental deadlines.

An undergraduate senior paper may or may not include original research, but must present an in-depth exploration of a topic in computer science (with particular focus on understanding and evaluating some element of the computer science literature). The paper should demonstrate the student's ability to apply, in a new context, the fundamental themes and objectives that connect all computer science classes, such as:

- separating a problem definition from its solution.
- describing clearly a proposed solution (typically with examples).
- understanding the correctness and applicability of a proposed solution.
- comparing several proposed solutions in terms of clarity, resource requirements, etc.

It is common for the thesis to center on a particular algorithm or computing system, and present the correctness and/or computational complexity thereof. However, this is not required. Students have successfully pursued other topics, such as human-computer interaction. The one core requirement is that the student demonstrates the ability to think deeply and communicate clearly about a computer science topic beyond the depth covered in classes.

The written thesis often resembles a review article, which explores in depth a collection of primary source articles from a single research group, or a survey article, which compares primary source articles from different origins.

The oral presentation is given after the thesis has been completed, though preliminary presentations are often also given as practice (and for formative assessment) during the year. The presentation is not graded, although all students are required to give one.

The learning goals for the research that goes into all of this are as follows:

Aspirational (for the best students):
A substantial written contribution that demonstrates original thinking and/or insight about a research area inside computer science, under the supervision of a faculty member. This should include a full literature review, appropriate replication of existing work, and either:

- a clear hypothesis (model), validation (proof/experiments), and analysis; *or*
- original expository work, including the extension of a proof, or a new proof of an existing theorem.
- Since such theses include original material, they may constitute part of a publication (typically a joint publication with the advisor). However, publication is *not* required.

Achievable (for most students):
A confirmation and reiteration of existing work with an incremental contribution. Specifically, this includes a full literature review and either:

- a good and complete confirmation of an existing experiment on new data, including a good analysis; *or*
- an exposition of non-trivial graduate-level published work, including an existing proof or deep explanation of its extension/applicability (or its lack of extension) to other related concepts.

Required (of all students):
A non-trivial literature review/exposition of

existing graduate-level published work, specifically:

The introductory material must be:

- readable by someone who has understood only the core CMSC undergrad material (e.g. programming languages, hardware, theory, algorithms, and at least one intensive systems course such as compilers or O.S.).
- detailed enough to be clear to someone within the field.

The discussion of related work should:

- include all the important related/foundational work.
- clearly identify what problem is being addressed by each work (possibly one statement of this for many/all the works).
- clearly state the basic approach being taken.
- explain how each paper supports/evaluates its own results (proof/empirical-study/ad-hoc argument).
- make clear how this work relates to the thesis itself.
- in at least one case, really address the details of how the approach works (possibly several such discussions will be needed to address the point above).

Senior Thesis Assessment

The grade is approximately 75% based on the work done under the supervision of the faculty advisor and about 25% based on meeting the deadlines of and participating in the senior seminar, including the fall poster and spring presentation.

The senior paper is primarily assessed by the student's advisor. Usually one or more other members of the department also read the paper and provide feedback for the student and advisor. If the student has a separate subject-matter advisor at another institution, that advisor is consulted during the grading of the paper if at all possible. All faculty involved in the thesis (and many students) are typically in attendance for the oral presentation.

The grade for the senior experience is assigned by the advisor, based on the quality of the student's written paper (judged in terms of illustrating mastery of the learning objectives relevant to the chosen topic), on participation in the oral presentation, and on the work habits illustrated during the year's work.

After thorough discussion by the department, a student's grade on the thesis will reflect how closely they have met the qualitative goals stated above. Specifically:

- 4.0: meets aspirational goals stated above.
- 3.0: meets achievable goals stated above.
- 2.0: meets required goals stated above.

All students should reach at least a 2.0 level of work on the material they submit by the end of the fall semester, and the faculty will certify students as having achieved this level (or not) in January.

In addition to submitting the written thesis document, students must also complete the assigned *presentation* elements, which typically include a December poster presentation of the thesis topic and scope, and the final oral presentation of the thesis. These presentations are graded on evidence of preparation and on participation (i.e. showing up on time for one's own presentation, attending the rehearsals of a few others, and providing feedback and/or asking questions). Faculty will provide informal feedback to the presenters on speaking style, professionalism, diction/grammar, poise, etc., but these elements are not included in the grade.

RELATED CONCENTRATION

Concentration in Scientific Computing

Computation is the *object of study* for the computer science major and minor; computation is also an important *tool* with which to study many other disciplines. The Concentration in Scientific Computing focuses on the application of computational techniques in other natural and social sciences.

For more information about the concentration, please see the program's website (*haverford.edu/scientific-computing*) or catalog entry.

AFFILIATED PROGRAM

Engineering

Computer science majors may pursue various engineering disciplines via our partnerships with the University of Pennsylvania and CalTech. More information on this partnership can be found on the departmental website (*www.haverford.edu/engineering*).

FACILITIES

Information on all hardware and software resources for the programs in computer science may be found at *haverford.edu/computer-science/resources*.

FACULTY

At Haverford:

Siddharth Bhaskar
Visiting Assistant Professor

John Dougherty
Associate Professor

Sorelle Friedler *(on leave 2017-2018)*
Assistant Professor

Steven Lindell
Professor

Suzanne Lindell
Laboratory Instructor

Kristopher Micinski
Visiting Assistant Professor

David Wonnacott
Chair and Associate Professor

Affiliated Faculty:

Lynne Butler
Professor of Mathematics

Jane Chandlee
Assistant Professor of Linguistics

Curtis Greene *(on leave 2017-2018)*
J. McLain King Professor of Mathematics

Robert Manning
William H. and Johanna A. Harris Distinguished Professor of Computational Science
Professor of Mathematics and Statistics

Joshua Schrier
Associate Professor of Chemistry

At Bryn Mawr:

Douglas Blank
Associate Professor

Richard Eisenberg
Assistant Professor

Deepak Kumar
Professor

Dianna Xu
Professor

COURSES

CMSC H104 TOPICS IN INTRODUCTORY PROGRAMMING

Jane Chandlee, Sorelle Friedler, Joshua Schrier
Natural Science (NA), Quantitative (QU)
Topics in Introductory Programming is designed to give a general introduction to programming as related to data analysis across many fields. Students will be introduced to standard introductory programming aproaches (e.g., imperative and object-oriented) as well as data structures necessary to create efficient and understandable algorithmic solutions to problems. Data for analysis will be drawn from a single discipline that will vary per semester, forming a theme for topical study. Topical investigations will include the ethics of data use in that field, how data is commonly generated and used, and implementation of important discipline-specific algorithms. Prerequisite(s): May not be taken by students who (a) have AP credit in Computer Science; or (b) have taken any one of HC: CMSC 105, CMSC 106, CMSC 107; BMC: CMSC 110, except by instructor consent. (Typically offered every spring)

CMSC H105 INTRODUCTION TO COMPUTER SCIENCE

John Dougherty
Natural Science (NA), Quantitative (QU)
Introduction to the intellectual and software tools used to create and study algorithms: formal and informal problem specification; problem solving and algorithm design techniques; reliability, formal verification, testing, and peer code review techniques; program clarity, complexity and efficiency; functional and imperative paradigms; associated programming skills. Students must attend a one-hour weekly lab. Labs will be sectioned by course professor. Prerequisite(s): May not be taken by students who have taken any one of HC: CMSC 104, CMSC 107; BMC: CMSC 110, except by instructor consent. (Offered Fall 2017)

CMSC H106 INTRODUCTION TO DATA STRUCTURES

John Dougherty
Natural Science (NA), Quantitative (QU)

An introduction to the fundamental data structures of computer science: strings, lists, stacks, queues, trees, BSTs, graphs, sets and their accompanying algorithms. Principles of algorithmic analysis and object reasoning and design will be introduced using mathematical techniques for the notions of both complexity and correctness. More practical issues, such as memory management and hashing, will also be covered. The programming language used to illustrate and implement these concepts will be able to support functional, imperative and object-oriented approaches. Emphasis will be placed on recursive thinking and its connection to iteration. Students must attend a one-hour weekly lab. Labs will be sectioned by course professor. Prerequisite(s): CMSC 105 (or 110 at Bryn Mawr) or instructor consent; may not be taken by students who have taken any one of HC: CMSC 104, CMSC 107; BMC: CMSC 206, except by instructor consent. (Typically offered every spring)

CMSC H107 INTRODUCTION TO COMPUTER SCIENCE AND DATA STRUCTURES
Siddharth Bhaskar, David Wonnacott
Natural Science (NA), Quantitative (QU)
An accelerated treatment of CMSC 105/106 for students with significant programming experience. Reviews programming paradigms, while focusing on techniques for reasoning about about software: methodical testing, formal verification, code reviews, other topics as time permits. Includes lab work. Prerequisite(s): CMSC104 or instructor consent, or placement by CS faculty, based on CS placement test. If you are interested in CMSC 107, you should preregister for the CMSC 105 section at the same time and take the placement test by the deadline, typically Wednesday before classes start; may not be taken by students who have taken any one of HC: CMSC 105, CMSC 106; BMC: CMSC 206, except by instructor consent. (Offered Fall 2017)

CMSC H207 DATA SCIENCE AND VISUALIZATION
Sorelle Friedler
Natural Science (NA)
An introduction to techniques for the automated and human-assisted analysis of data sets. These "big data" techniques are applied to data sets from multiple disciplines and include cluster, network, and other analytical methods paired with appropriate visualizations. Prerequisite(s): CS105 *and* 106 *or* CS107 *or* instructor consent. (Not offered 2017-18)

CMSC H208 SPEECH SYNTHESIS AND RECOGNITION
Jane Chandlee
Natural Science (NA)
An introduction to the methodologies used in the automated recognition and synthesis of human speech, focusing on Hidden Markov Models in recognition and unit selection in synthesis. Students will get hands-on experience with implementing the various components of these systems. Crosslisted: Computer Science, Linguistics; Prerequisite(s): CS105 and 106 OR CS107 OR BMC 110 and 206 OR instructor consent. (Offered Fall 2017)

CMSC H210 LINEAR OPTIMIZATION AND GAME THEORY
Curtis Greene
Natural Science (NA)
Covers in depth the mathematics of optimization problems with a finite number of variables subject to constraints. Applications of linear programming to the theory of matrix games and network flows are covered, as well as an introduction to nonlinear programming and hidden Markov models. Emphasis is on the structure of optimal solutions, algorithms to find them, and the underlying theory that explains both. This course is designed for students interested in computer science, economics, or mathematics. Prerequisite(s): MATH 215 or equivalent or instructor consent. (Not offered 2017-18)

CMSC H215 HUMAN COMPUTER INTERACTION
John Dougherty
Natural Science (NA)
Covers the design, evaluation and implementation of interactive computing systems, along with the study of major phenomena surrounding these systems. Topics include: user-centered design, usability, affordances, cognitive and physical ergonomics, information and interactivity structures, interaction styles, interaction techniques, and user interface tools with a special focus on accessible and mobile interfaces. Prerequisite(s): CMSC106, 107, 206, or instructor consent. (Offered Spring 2018)

CMSC H222 SCIENTIFIC COMPUTING: CONTINUOUS SYSTEMS
Robert Manning
Natural Science (NA), Quantitative (QU)
A survey of major algorithms in modern scientific computing, with a focus on continuous problems. Topics include numerical differentiation and integration, numerical linear algebra, root-finding, optimization, Monte Carlo methods, and discretization of differential equations. Basic ideas of error analysis are presented. A regular computer lab introduces students to the software package Matlab, in which the algorithms are implemented and applied to various problems in the natural and social sciences. Crosslisted: Mathematics, Computer Science; Prerequisite(s): MATH 215 or instructor consent. (Offered Fall 2017)

CMSC H231 DISCRETE MATHEMATICS
Siddharth Bhaskar, Steven Lindell
Natural Science (NA)
An introduction to discrete mathematics with strong applications to computer science. Topics include set theory, functions and relations, propositional logic, proof techniques, difference equations, graphs, and trees. Crosslisted: Computer Science, Mathematics; Co-requisite(s): CMSC 105, 107, or 110, or instructor consent. (Offered Fall 2017)

CMSC H240 PRINCIPLES OF COMPUTER ORGANIZATION
John Dougherty
Natural Science (NA)
Treatment of the hierarchical design of modern digital computers: boolean logic/algebra; truth tables; combinational and sequential circuits; state systems; register machines; instruction sets; memory organization; assembly language programming. Lectures cover the theoretical aspects of system architecture; labs provide implementation experience via a hardware simulator. Concurrent enrollment in this and two other CMSC lab courses requires permission of the instructor. Prerequisite(s): CMSC 106, or 107 or 206, and CMSC/Math 231 (or instructor consent). (Typically offered every fall)

CMSC H245 PRINCIPLES OF PROGRAMMING LANGUAGES
Kristopher Micinski, David Wonnacott
Natural Science (NA)
Study of the design and implementation of modern programming languages: lexical and syntactic analysis; scoping mechanisms; run-time environments; implementation of structured, functional, object-oriented, and concurrent programming languages. Lectures cover theoretical foundations of language design and implementation; labs provide opportunities to both use and implement language features. Concurrent enrollment in this and two other CMSC lab courses requires permission of the instructor. Prerequisite(s): CMSC 106, or 107 or 206, and CMSC/Math 231 (or instructor consent). (Offered Fall 2017)

CMSC H287 HIGH PERFORMANCE SCIENTIFIC COMPUTING
John Dougherty
Natural Science (NA)
Introduction to parallel and distributed systems and approaches found in scientific computing, including computational and data intensive applications. Primary lab work on a cluster of Linux workstations with C, OpenMP, and MPI; other architectures and approaches are also covered. Prerequisite(s): CMSC 106 or instructor consent. (Typically offered every other year)

CMSC H325 COMPUTATIONAL LINGUISTICS
Jane Chandlee
Natural Science (NA)
An overview of key areas of computational linguistics, including natural language processing and computational modeling of morpho-phonological systems. Students will study and practice the primary algorithms and techniques used in the automated analysis of natural language data. Crosslisted: Computer Science, Linguistics; Prerequisite(s): CMSC 105 and CMSC 106 (or CMSC 107), OR CMSC B110 and CMSC B206, OR instructor consent. (Not offered 2017-18)

CMSC H340 ANALYSIS OF ALGORITHMS
Sorelle Friedler, Steven Lindell
Natural Science (NA)
Qualitative and quantitative analysis of algorithms and their corresponding data structures from a precise mathematical point of view. Performance bounds, asymptotic and probabilistic analysis, worst case and average case behavior. Correctness and complexity. Particular classes of algorithms such as sorting searching will be studied in detail. Crosslisted: Mathematics, Computer Science; Prerequisite(s):

CMSC 106 or 107 or B206, and 231, or instructor consent. (Typically offered every fall)

CMSC H345 THEORY OF COMPUTATION
Steven Lindell
Natural Science (NA)
Introduction to the mathematical foundations of computer science: finite state automata, formal languages and grammars, Turing machines, computability, unsolvability, and computational complexity. Attendance at the discussion section on Friday is required. Crosslisted: Mathematics, Computer Science; Prerequisite(s): (CMSC 106 or CMSC 107) and CMSC 231, and junior or senior standing, or instructor consent. (Typically offered every spring)

CMSC H350 COMPILER DESIGN
David Wonnacott
Natural Science (NA)
An introduction to compiler design, including the tools and software design techniques required for compiler construction. Students construct a working compiler using appropriate tools and techniques in a semester-long laboratory project. Lectures combine practical topics to support lab work with more abstract discussions of software design and advanced compilation techniques. Concurrent enrollment in this and two other CMSC lab courses requires permission of the instructor. Prerequisite(s): CMSC 245 or instructor consent. (Offered Spring 2018)

CMSC H356 CONCURRENCY AND CO-DESIGN IN OPERATING SYSTEMS
John Dougherty, David Wonnacott
Natural Science (NA)
A practical introduction to the principles of shared-memory concurrent programming and of hardware/software co-design, which together underlie modern operating systems; includes a substantial laboratory component, currently using Java's high-level concurrency and the HERA architecture. Concurrent enrollment in this and two other CMSC lab courses requires permission of the instructor. Prerequisite(s): CMSC 240 or instructor consent. (Typically offered every other spring)

CMSC H399 SENIOR THESIS
Steven Lindell
Natural Science (NA)
Fall seminar required for seniors writing theses, dealing with the oral and written exposition of advanced material. (Offered Fall 2017)

CMSC H480 INDEPENDENT STUDY
Staff
Natural Science (NA)
Independent study, supervised by a member of the Computer Science department.
Prerequisite(s): Instructor consent. (Offered occasionally)