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 emphasizes these fundamental concepts of the field, with emphasis on depth of thought and clarity of expression. 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**

Specific learning objectives are for each student 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 an specific 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 his or her 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 his or her 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.

**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.

**Major**

The major in computer science is designed for students who wish to explore fundamental questions about computation itself 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.

**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 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 major 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).
- CMSC 240 (Principles of Computer Organization).
- CMSC 245 (Principles of Programming Languages).
- CMSC 340 (Analysis of Algorithms).
- CMSC 345 (Theory of Computation).
- Either CMSC 240 (Principles of Computer Organization) and a course on operating systems, 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 345 (Theory of Computation).

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, 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 345 (Theory of Computation).

CONCENTRATION REQUIREMENTS
The Computer Science Department supports the Concentration in Scientific Computing, available to a variety of majors, 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 additional 300-level computer science course.
- 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 PROJECT
The senior thesis in computer science is a year-long research experience under the guidance of a faculty member that culminates in a written paper and an oral presentation. The research experience can include original work, but it must demonstrate original exposition and thinking, always including a thorough literature review.

To ensure that students successfully complete this graduation requirement, they are required to enroll in a year-long one credit senior seminar course in the department, CMSC 399. In this course, there

Haverford College Catalog 2016-2017
are a series of class activities and deadlines to help keep students on track for completing their senior thesis. In the fall semester, these include: the adviser selection process; submitting the topic proposal; completing the literature review; the public poster presentation; and the chapter one thesis draft. In the spring semester, these include: completing a rough draft of their thesis; rehearsing their oral presentations; submitting the final thesis document; and giving their oral presentation. In between the two semesters, a second reader provides feedback to the student and their adviser as to whether the 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 CMSC 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/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 adviser). 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,
COMPUTER SCIENCE

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 nuts and bolts of how the approach works (possibly several such discussions will be needed to address the point above).

Senior Project Assessment
The grade is approximately 75% based on the work done under the supervision of the faculty adviser 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 adviser. Usually one or more other members of the department also read the paper and provide feedback for the student and adviser; if the student has a separate subject-matter adviser at another institution, that adviser is consulted during the grading of the paper if at all possible. All faculty (and many students) are typically in attendance for the oral presentation.

The grade for the senior experience is assigned by the adviser, 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 uses of computational techniques in other natural and social sciences.

For more information about the Concentration and its requirements, 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.

FACILITIES
See the departmental web page for a description of laboratories, equipment and other special facilities for this program.
COMPUTER SCIENCE

FACULTY
At Haverford:
Steven Lindell
Professor

John P. Dougherty
Associate Professor

David G. Wonnacott
Associate Professor

Sorelle A. Friedler
Assistant Professor

Jane Chandlee
Visiting Assistant Professor

Suzanne Lindell
Laboratory Instructor

Affiliated Faculty:
Lynne Butler
Professor of Mathematics

Curtis Greene
J. McLain King Professor of Mathematics

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

Philip M. Meneely
Professor of Biology

Walter Smith
Professor of Physics

At Bryn Mawr College:
Deepak Kumar
Professor

Dianna Xu
Professor

Douglas Blank
Associate Professor

Richard Eisenberg
Assistant Professor

COURSES
CMSCH104 TOPICS IN INTRODUCTORY PROGRAMMING
John Dougherty, Sorelle Friedler

A general introduction to computer programming, in the context of its application to a specific discipline such as Data Analysis or Bioinformatics. Prerequisite for CMSC 107, along with discipline-specific analysis. Not for students who have completed CMSC 105 and/or 107. Quantitative (QU); Natural Science (NA)

CMSCH105 INTRODUCTION TO COMPUTER SCIENCE
Staff
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, proofs, and testing techniques; program clarity, complexity and efficiency; functional and imperative paradigms; associated programming skills. Weekly programming laboratory section. Quantitative (QU); Natural Science (NA)

CMSCH106 INTRODUCTION TO DATA STRUCTURES
John Dougherty
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. Some 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 Python, and emphasis will be placed on recursive thinking and its connection to iteration. Labs will be sectioned by course professor. Prerequisite(s): CMSC 105 (or 110 at Bryn Mawr) or consent. Students must attend one, one-hour lab per week. Quantitative (QU); Natural Science (NA)

CMSCH107 INTRODUCTION TO COMPUTER SCIENCE AND DATA STRUCTURES
David Wonnacott
An accelerated treatment of CMSC 105/106 for students with significant programming experience. Reviews programming paradigms, while focusing on techniques for reasoning about software: methodical testing, formal verification, code reviews, other topics as time permits. Includes lab work. Fulfills CCNC concentration requirement. Prerequisite(s): placement by CMSC faculty, based
on CMSC placement test. If you are interested in CMSC 107, you should preregister for the CMSC 105 section at the same time, take the placement test by the deadline (typically Wednesday before classes start). Quantitative (QU)

CMSCH207 DATA SCIENCE AND VISUALIZATION
Sorelle Friedler
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): (CMSC 105 and CMSC 106) or CMSC 107 and CMSC 231 or permission of the instructor; Natural Science (NA)

CMSCH208 SPEECH SYNTHESIS AND RECOGNITION
Jane Chandlee
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. Prerequisite(s): CMSC 105 and 106 OR CMSC 107 OR BMC 110 and 206 OR instructor permission; Natural Science (NA)

CMSCH231 DISCRETE MATHEMATICS
Steven Lindell, Jane Chandlee
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. Corequisite(s): CMSC 105, 107, or 110; Crosslisted: MATH; Natural Science (NA)

CMSCH240 PRINCIPLES OF COMPUTER ORGANIZATION
John Dougherty
Treatment of the hierarchical design of modern digital computers: boolean logic/algebra; sequential 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. Prerequisite(s): CMSC 106, 107, or B206 or consent. CMSC/Math 231 strongly recommended. Concurrent enrollment in this and two other CMSC lab courses requires permission of the instructor. Natural Science (NA)

CMSCH245 PRINCIPLES OF PROGRAMMING LANGUAGES
John Dougherty, David Wonnacott
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. Prerequisite(s): CMSC 106, 107, or B206 or consent. CMSC/MATH 231 strongly recommended. Concurrent enrollment in this and two other CMSC lab courses requires permission of the instructor. Natural Science (NA)

CMSCH287 HIGH-PERFORMANCE SCIENTIFIC COMPUTING
John Dougherty
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 MPI; other architectures and approaches are also covered. Prerequisite(s): CMSC 106 or consent; Natural Science (NA)

CMSCH300 COMPUTER SCIENCE RESEARCH FOUNDATIONS
Staff
An introduction to research skills needed for the field of computer science, designed to prepare students for senior thesis or summer research work. Natural Science (NA)

CMSCH306 RELATIONAL DATABASE DESIGN AND UTILIZATION
Staff
Relational Database Design and Utilization covers the principles of relational database design, including some complex corner cases and solutions to scaling issues with the technology. The course will utilize MySQL as the exemplar relational database type, but will discuss key differences with other implementations. The Django programming framework will be employed both for expedience and as a common practical utility in database design. The course will also include an investigation study on other forms of relational and non-
COMPUTER SCIENCE

relational database types, and a case study on a complex relational database design. Prerequisite(s): CMSC 106 or CMSC 107; Natural Science (NA)

CMSCH340 ANALYSIS OF ALGORITHMS
Sorelle Friedler
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. Prerequisite(s): CMSC 106 or 107 or B206, and 231; Natural Science (NA)

CMSCH345 THEORY OF COMPUTATION
Steven Lindell
Introduction to the mathematical foundations of computer science: finite state automata, formal languages and grammars, Turing machines, computability, unsolvability, and computational complexity. Attendance required. Prerequisite(s): (CMSC 106 or CMSC 107) and CMSC 231 or consent; Natural Science (NA)

CMSCH350 COMPILER DESIGN
David Wonnacott
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. Prerequisite(s): CMSC 245; concurrent enrollment in this and two other CMSC lab courses requires permission of the instructor; Natural Science (NA)

CMSCH356 CONCURRENCY AND CO-DESIGN IN OPERATING SYSTEMS
David Wonnacott
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. Prerequisite(s): CMSC 240; concurrent enrollment in this and two other CMSC lab courses requires permission of the instruction. Natural Science (NA)

CMSCH399 SENIOR THESIS
Staff
Natural Science (NA)

COURSES OFFERED AT BRYN MAWR
CMSCB110 INTRODUCTION TO COMPUTING
CMSCB206 INTRODUCTION TO DATA STRUCTURES
CMSCB231 DISCRETE MATHEMATICS
CMSCB246 PROGRAMMING PARADIGMS
CMSCB355 OPERATING SYSTEMS
CMSCB380 RECENT ADVANCES IN COMPUTER SCIENCE
CMSCB312 COMPUTER GRAPHICS
CMSCB371 COGNITIVE SCIENCE