Proposal for Revision of the Computer Science Curriculum
Drew University

Shannon Bradshaw[1], Steve Kass, Barry Burd, Fran Trees, and Chris Van Wyk

1 Rationale

Benchmark: The Association for Computing Machinery (ACM) is the principal organization of computer scientists and members of computing-related fields. Once per decade, the ACM releases a report on undergraduate computing curriculum and recommendations for programs that grant degrees in computer science. The committees generating such reports are composed of computer science educators from across the discipline at research universities, liberal arts colleges, and other types of schools. Recent recommendations were the product of teams including notables such as Eric Roberts (Stanford) and Kim Bruce (formerly at Williams College, now at Pomona). ACM curriculum committees base their recommendations on extensive surveys of existing computer science programs and thoughtful consideration of changes in technology and culture that prompt the need for changes in undergraduate education. Computer science programs take ACM's recommendations seriously and attempt to keep their curriculum in line with the most current guidelines.

Meeting the CC2001 core requirements: The most recent computing curricula recommendations were published by ACM in 2001 (CC2001) [4]. CC2001 organizes the computer science body of knowledge into 14 areas of specialty within the discipline. Within each area it identifies a subset of topics that are essential material. The essential material from each area constitute a minimal core curriculum for computer science undergraduate education. As stated in CC2001, the core consists of "material that essentially everyone teaching computer science agrees is essential to anyone obtaining an undergraduate degree in this field." Drew's current computer science curriculum is more closely aligned with CC1991 [7]. The proposed curriculum corrects this problem, fully incorporating all CC2001 core material.

Meeting the CC2001 broader recommendations: By definition, several of our peer schools meet many of the CC2001 core requirements. However, few address the primary recommendations that set CC2001 apart from CC1991. These recommendations represent a significant departure from the vertically-oriented and theoretical nature of most undergraduate computer science programs. CC2001 reflects an increased emphasis on a broad base of applied material, with special attention to topics such as software engineering, professional practice, and human factors (see pp. 62-66 of CC2001). These topics draw on the fields of psychology, sociology, economics, and art and due to their interdisciplinary nature are well-suited to the liberal arts environment.

Drew is fortunate in that it is a liberal arts college whose computer science faculty work in applied areas of the discipline specifically emphasized by CC2001. In addition, Drew is well-positioned in its proximity to a major industrial center. Finally, most of our computer science graduates pursue careers in industry following graduation [3, 2, 1]. In general, most computer science students choose industry over graduate school (a ratio of 6:1) after earning their bachelor's degree [5]. This statistic exemplifies a striking difference between computer science and the natural and social sciences.

[1]sbradsha@drew.edu

CC2001, the expertise of our faculty, and the behavior of Drew computer science graduates and that of computer science graduates in general all point to an applied curriculum for Drew's Computer Science program. Therefore, we have selected from among the large body of CC2001 applied computing topics those that we feel are most in keeping with where the discipline is now and where it will be in two to five years.

Preparing students for graduate school: The line between research and industry in computing is not well-defined. Obvious examples include IBM, Amazon, and Google. But smaller companies too consume and contribute to the body of computer science scholarship. See www.inxight.com, www.fogcreek.com, and www.alias-i.com. For this reason and in the interest of currency, it is necessary to include scholarly literature as at least part of the material for several courses. As part of our new curriculum, many of the upper level courses will require students to read research papers germane to the subject (e.g., CSCI 130, CSCI 150, CSCI 160). They will also be required to study existing solutions in a particular problem space. Our objective is to stimulate an understanding of the process of innovation. The proposed emphasis on problem analysis and solution design in a variety of areas will prepare our students well for both industry and a range of graduate programs.

Inter-disciplinary considerations: We have taken steps to ensure the ability to address computing needs across the disciplines at Drew in the proposed curriculum. At least some of the material in every course is of an interdisciplinary nature. Particularly noteworthy examples include, CSCI 10 and CSCI 160. We hope to find ways of reaching out to other disciplines from these courses. In addition, we envision an inter-disciplinary computational studies minor (see [6]) merging computing with one or more application areas. Finally, we are working toward consensus among mathematics, computer science, the natural sciences, and social sciences to define a computational foundation for students across these disciplines.

Staffing: This proposal imposes no need for additional instructors.

2 Changes to the Computer Science Curriculum

The proposed changes to the computer science curriculum are described in Table 1. Every course that we propose dropping from the curriculum is composed of much material not considered essential by CC2001. The CC2001 essential material from the dropped courses has been reorganized and integrated into courses we propose either keeping or adding. As described above, the newly added courses reflect our objective to keep pace with the discipline of computing.

3 Impact on Other Departments

Linguistics: The Linguistics Studies minor requirements list CSCI 105, 107, 126 among fourteen elective courses from which students must choose eight hours. While there are no clear alternatives to these courses, some of the content of CSCI 105 has been incorporated into CSCI 101. Other options would include CSCI 1, CSCI 2, and CSCI 10, each of which includes treatment of a programming language.

Offerings			Actions				
Current	Proposed	Title	Drop	Add	Renumber	Change title or description	None
CSCI 1	CSCI 1	Introduction to Computers and Computing					CSCI 1
CSCI 2	CSCI 2	Object-Oriented Programming					CSCI 2
	CSCI 10	Human Interaction with Technology and Information		CSCI 10			
CSCI 23	CSCI 23	Discrete Mathematics for Computer Science					CSCI 23
CSCI 25	CSCI 25	Data Structures					CSCI 25
CSCI 27 (2 cr) •		Social and Ethical Issues in Computing Technology	CSCI 27				
CSCI 70	CSCI 70	Computing Technology, Society, and Culture					CSCI 70
	CSCI 100	Systems Programming and Tools		CSCI 100			
CSCI 101	CSCI 101	Algorithm Analysis and Computability (was Algorithms)				CSCI 101	
CSCI 103 •		Computer Organization	CSCI 103				
CSCI 105 •		Computability	CSCI 105				
CSCI 107		Programming Languages	CSCI 107				
CSCI 124	CSCI 124	Operating Systems					CSCI 124
CSCI 126 •		Artificial Intelligence	CSCI 126				
CSCI 128 •		Foundations of Computer Graphics	CSCI 128				
CSCI 141*	CSCI 130	Information Management *when topic was databases		CSCI 130			
CSCI 133		Numerical Methods	CSCI 133				
CSCI 141*	CSCI 140	Software Engineering *when topic was software design		CSCI 140			
	CSCI 150	Net-Centric Computing		CSCI 150			
	CSCI 160	Applications of Computing to Other Disciplines		CSCI 160			
CSCI 171		Seminar in Computer Science	CSCI 171				
CSCI 141* •	CSCI 198	Topics in Computer Science *other topics not mentioned above			141→198		
CSCI 155	CSCI 199	Independent Study in Computer Science			155→199		

Legend
most content dropped
some content moved
modified
unchanged
new

$\bullet\!\longrightarrow$ Specific topics have been moved to another course

Table 1: Summary of proposed changes to the CS Curriculum.

Economics: The Economics Department recommends CSCI 2 to their students who plan to pursue graduate study. This course remains in the proposed curriculum; however, we are proposing an additional prerequisite of CSCI 1.

Chemistry: Under the three-two chemical engineering option of the chemistry major, CSCI 2 is listed as a required course. CSCI 2 remains in the proposed curriculum; however, we are proposing an additional prerequisite of CSCI 1.

Neuroscience: Neuroscience lists CSCI 2 and CSCI 25 among the choices for Section III of the major. These courses remain in the proposed curriculum; however, the prerequisite path for both courses now includes CSCI 1. If the intent of this requirement is that students develop a solid programming foundation, we recommend modifying the requirement to CSCI 1 and CSCI 2.

The neuroscience requirements also list CSCI 101 and CSCI 126 among the elective options. CSCI 101 has much of the same content under the proposed curriculum as it does in the existing curriculum; however, it includes content from CSCI 105, and one topic from CSCI 126. With the exception of this one topic, the material from CSCI 126 has been entirely removed in the proposed curriculum.

Political Science: CSCI 1 is strongly recommended for political science majors. This course remains in the proposed curriculum with no changes.

Theatre Arts: The requirements for the Arts Administration and Museology with Emphasis in Theatre minor list CSCI 70 as an elective. CSCI 70 will remain in the proposed curriculum and is unchanged.

4 Computer Science Major

4.1 Proposed Requirements

The proposed revised major consists of 13 required courses. No electives are required, though elective material will be made available to students through CSCI 70, CSCI 198, and CSCI 199. Each course is a four credit hour offering. A total of 52 hours will be required to complete the Computer Science major. This represents an increase of four hours over existing major requirements. The proposed major requirements are as follows:

- CSCI 1: Introduction to Computers and Computing
- CSCI 2: Object-Oriented Programming
- MATH 3: Introduction to Statistics
- CSCI 10: Human Interaction with Technology and Information
- CSCI 23: Discrete Mathematics for Computer Science
- CSCI 25: Data Structures
- CSCI 100: Systems Programming and Tools
- CSCI 101: Algorithm Analysis and Computability
- CSCI 124: Operating Systems
- CSCI 130: Information Management
- CSCI 140: Software Engineering
- CSCI 150: Net-Centric Computing
- CSCI 160: Capstone - Applications of Computing to Other Disciplines

4.2 Course Descriptions

CSCI 1: Introduction to Computers and Computing: (Existing) An introduction to problem solving with computers. Tools for problem solving include the Alice 3D Authoring System, Adobe Flash, and other graphical application building environments.

CSCI 2: Object-Oriented Programming: (Existing) Designing, writing, and testing structured computer programs. Decomposing problems; writing function definitions; conditional and iterative control constructs; using class libraries. Problem-solving through programming with classes and vectors; algorithm correctness; recursion. Java will be the language of instruction. *Prerequisite: C- or better in CSCI 1*

CSCI 10: Human Interaction with Technology and Information: (New) A study of how people perceive technology and the ways in which they consume and create information. An introduction to the practice of designing technology with careful consideration for its users. No programming experience is required prior to taking this course.

CSCI 23: Discrete Mathematics for Computer Science: (Existing) Mathematics central to the study of computer science. Topics include: set theory, logic, induction, combinatorics, number theory, graph theory, sequences and series, matrices, and recurrence relations. *Prerequisite: C- or better in CSCI 1*

CSCI 25: Data Structures: (Existing) Introduction to the study of abstract data types and the analysis of algorithms. Students will write Java applications using data structures such as linked lists, stacks, queues, multidimensional arrays, trees, sets, maps, and heaps. *Prerequisite: C- or better in CSCI 2*

CSCI 27: Computers: Social and Ethical Issues: (Discontinued) An investigation of social and ethical issues related to computers and computing technologies, emphasizing current events and controversies. Topics will include privacy, security, the First and Fourth Amendments, computer crime, and corporate responsibility. Enrollment limit: 18.

CSCI 70: Computing Technology,Society, and Culture: (Existing) This course will survey the principal computing technologies that are in use today or on the horizon, then investigate individual topics in more technical and cultural depth. Topics will vary in light of new developments, and could include blogging, RFID, intelligent systems, GPS, data mining, Google, and eBay. Other aspects of computing technology, society, and culture to be addressed will include legal and political issues such as regulation, jurisdiction, internationalization, and standardization, and broader questions such as how and why new computing technologies are developed and accepted. Enrollment priority: Priority is given to Juniors and Seniors. *Prerequisite: Sophomore standing required.*

CSCI 100: Systems Programming and Tools: (New) Development of software in the C programming language. User-level functionality of the UNIX operating system. Architecture of the UNIX operating system from a programmer's perspective. Machine-level representation of data; assembly-level machine organization. Tools for large-scale software engineering including integrated development environments and code versioning systems. *Prerequisite: C- or better in CSCI 2*

CSCI 101: Algorithm Analysis and Computability: (Modified) Methods for the analysis of time and space efficiency, comparison of brute-force algorithms with divide-and-conquer algorithms, tree algorithms, graph algorithms, string algorithms, dynamic programming, and greedy methods. An introduction to NP-completeness and intractability. Turing machines, Church's thesis, determinism and non-determinism, unsolvability and reducibility. Search and constraint satisfaction. *Prerequisite: CSCI 23 and CSCI 25.*

CSCI 103: Computer Organization: (Discontinued) Representations of numbers and characters, logic design, memory organization, microprogramming, pipelines, assembly language and translation, loading, and relocation, arithmetic, non-von Neumann architectures. *Prerequisite: C- or better in CSCI 2 and either C- or better in CSCI 25 or concurrent registration in CSCI 25.*

CSCI 105: Computability: (Discontinued) Automata, languages, Turing machines, Church's thesis, determinism and nondeterminism, recursive function theory, reducibility among problems, NP-completeness, unsolvability. *Prerequisite: C- or better in either CSCI 23 or MATH 100.* Same as: MATH 105.

CSCI 107: Programming Languages (Discontinued) Common features of programming languages, with attention to von Neumann languages; functional languages, data-abstraction languages, object-oriented languages, and logic programming languages. *Prerequisite: C- or better in CSCI 25.*

CSCI 124: Operating Systems: (Modified) The fundamentals of operating systems design and implementation. Basic structure; synchronization and communication mechanisms; implementation of processes, process management, scheduling, and protection; memory organization and management; file systems; machine-level representation of data; assembly-level machine organization; functional organization of computers. *Prerequisite: C- or better in CSCI 100.*

CSCI 126: Artificial Intelligence: (Discontinued) The study of attempts to program machines to behave intelligently. Examples of topics include production systems, knowledge representation, heuristic search, resolution theorem proving, natural language processing, and visual perception. *Prerequisite: C- or better in CSCI 107.*

CSCI 128: Foundations of Computer Graphics: (Discontinued) Mathematical foundations and algorithms for representing, rendering, manipulating, and interacting with two- and three-dimensional images on computer monitors. Implementations aided by graphics class libraries. Recommended: MATH 103 or CSCI 23. *Prerequisite: C- or better in both CSCI 25 and MATH 17.*

CSCI 133: Numerical Methods: (Discontinued) See Mathematics listing for course description. *Prerequisite: C- or better in CSCI 2 and in MATH 8.* Same as: MATH 133.

CSCI 130: Information Management: (New) Theory and practice of information storage, management and retrieval, emphasizing relational database management systems. Case studies of small-scale (personal computing) and large-scale (corporate records on distributed systems) applications. Data modeling, database design and management, query processing, data integrity, and security. Legal and social contexts of data management; the responsibility of professionals to understand requirements, risks, and liabilities. *Prerequisite: C- or better in CSCI 2 and CSCI 10.*

CSCI 140: Software Engineering: (New) Software design; using APIs; software tools and environments; software processes; software requirements and specifications; software validation; software evolution; software project management; methods and tools of working in teams; social context of computing; professional and ethical responsibilities; risks and liabilities of computer-based systems. *Prerequisite: C- or better in MATH 3, CSCI 10 and CSCI 100.*

CSCI 150: Net-centric Computing: (New) Communication and networking; the social context of computing; intellectual property; network security; the web as an example of client-server computing; building web applications; network management; compression and decompression; wireless and mobile computing; virtual machines; knowledge representation and reasoning. *Prerequisite: CSCI 23 and C- or better in MATH 3, CSCI 10, CSCI 100.*

CSCI 160: Capstone - Applications of Computing to Other Disciplines: (New) Much of computer science is practiced through application of computing to other disciplines. In this capstone course, the instructor and students will develop a software solution to a problem arising in another field. Application areas include, but are not limited to finance, economics, biology, and law. We will explore strategies for learning in and contributing to inter-disciplinary teams, customer-client communication; software design, requirements, specification, and project management. *Prerequisite: CSCI 23, CSCI 25, and C- or better in MATH 3, CSCI 10, and CSCI 100.*

CSCI 171: Seminar in Computer Science: (Discontinued) Topics to be chosen by the instructor. Work involves reading research articles, writing one or more papers, and making classroom presentations. May be repeated for credit with the approval of the department. *Prerequisite: Senior standing or signature of instructor.*

CSCI 198: Topics in Computer Science: (Renumbered) Topics to be determined by current events in computing and opportunities presented by visiting faculty, etc. Offered fall of even numbered years. *Prerequisite: Dependent on topic.*

CSCI 199: Independent Study in Computer Science: (Renumbered) An independent investigation of a topic selected in conference with the instructor and approved by the department. Admission by petition to or invitation from the department. Amount of credit established at time of registration. May be repeated for credit with the approval of the department. Signature of instructor required for registration.

4.3 Addressing Learning Objectives

The following describe the learning objectives the computer science faculty has for graduates of our program. They represent a revision of learning objectives submitted with the most recent Computer Science self-study in order to bring our objectives in line with CC2001 and a revised mission to be submitted with the Mathematics and Computer Science five-year plan in 2007. See CC2001, pp. 62-66 for a discussion of learning objectives for computer science graduates.

1. Foundational knowledge: Demonstrate knowledge and understanding of essential facts, concepts, principles, and theories relating to computer science and software applications.

2. Modeling and implementation: Deploy appropriate theory, practices, and tools for the specification, design, implementation, and evaluation of computer-based solutions.

3. Numeracy: Understand, explain, and determine the quantitative dimensions problems and systems.

4. Tools: Deploy effectively, information applications and systems, software development environments, and other tools commonly used in modern computing practice.

5. Evaluation and testing: Evaluate the extent to which a computer-based system meets the criteria defined for its current use and future development. Test systems in terms of general quality attributes, correctness, and possible tradeoffs presented within the given problem.

6. Requirements Analysis: Identify and analyze criteria and specifications appropriate to specific problems, and plan strategies for their solution.

7. Information management: Apply the principles of effective information management, information organization, and information-retrieval to information of various kinds (e.g., text and structured data).

8. Human-computer interaction: Apply the principles of human-computer interaction to the evaluation and construction of a wide range of materials including user interfaces, web pages, and multimedia systems.

9. Communication: Make succinct presentations to a range of audiences about technical problems and their solutions. Actively listen to customers, team members, and others with whom one works for effective exchange of information.

10. Professional development: Keep abreast of current developments in the discipline to continue one's own professional development.

11. Social context of computing: Demonstrate knowledge and understanding of the opportunities, implications, and risks of computing in a connected society and technologies that bear on each of these issues.

12. Professional responsibility: Recognize and be guided by the social, professional, and ethical issues involved in the use of computer technology. Manage one's own learning and development, including time management and organizational skills.

13. Project management: Be able to work effectively as a member of a development team. Knowledge and understanding of the entire software development lifecycle.

Table 2 summarizes the degree to which each course in the curriculum addresses each of the learning objectives. Each cell indicates the intensity with which a learning objective is addressed in a particular course. The scale used is zero to three stars, with three indicating a high level of coverage and zero indicating only incidental coverage.

4.4 Prerequisite Structure

Figure 1 summarizes the dependencies between courses in the proposed curriculum.

4.5 Schedule of Offerings

Table 3 describes the schedule on which the proposed curriculum will be offered. We have sufficient teaching resources to offer courses on this schedule.

	MATH	CSCI											
Learning Objective	**3**	**1**	**2**	**10**	**23**	**25**	**100**	**101**	**124**	**130**	**140**	**150**	**160**
Foundational knowledge	***	***	***		***	***	**	***	***	*		***	
Modeling and implementation	*	*	*	*	*	*	*	*	*	**	*	**	***
Numeracy	***	*	*		***	***		***		*	*	*	
Tools	**	*	*				***			*		*	
Evaluation and testing	*	*	*			*			*		***		
Requirements analysis				**		*		*			***		***
Information management				*			*			***		*	
Human-computer interaction				***			*			*		*	
Communication				**					*	*	*	*	*
Professional development							**		*	**		*	**
Social context of computing									**	*	*	***	*
Professional responsibility									*		*		*
Project management											***	*	*

Table 2: Addressing learning objectives in the proposed curriculum. Scale: zero to three stars. Three stars indicates a high level of coverage. Zero indicates only incidental coverage.

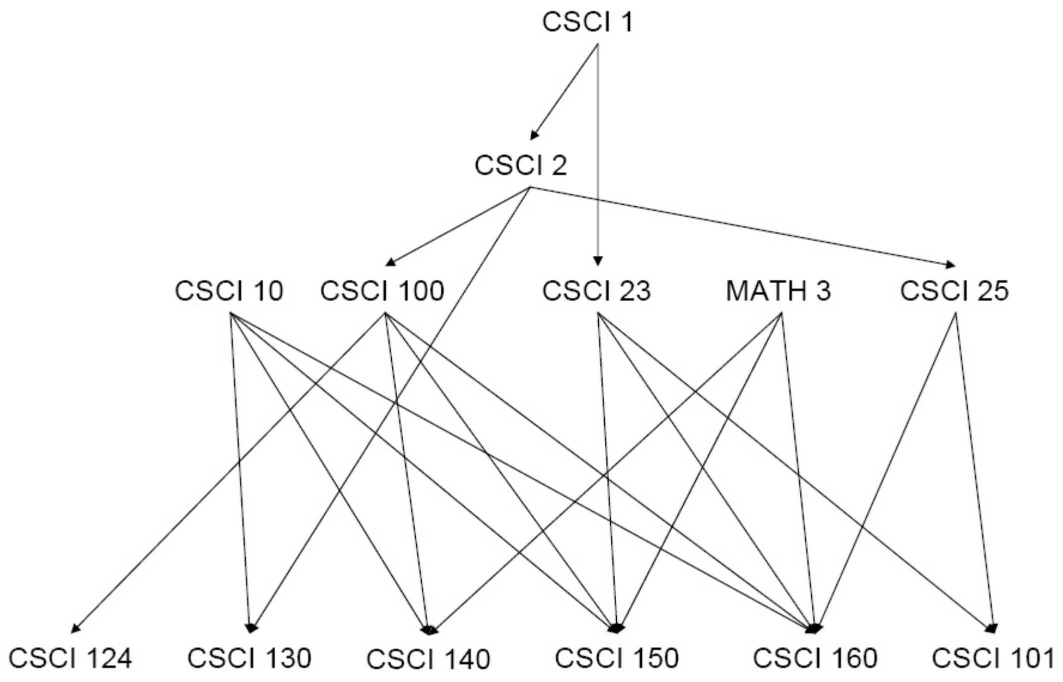

Figure 1: Course dependencies.

Course	Offered
CSCI 1: Introduction to Computers and Computing	fall and spring
CSCI 2: Object-Oriented Programming + **Lab**	fall and spring
CSCI 10: Human Interaction with Technology and Information	fall and spring
CSCI 23: Discrete Mathematics for Computer Science	fall
CSCI 25: Data Structures	spring
CSCI 70: Social and Ethical Issues in Computing Technology	spring
CSCI 100: Systems Programming and Tools	spring
CSCI 101: Algorithm Analysis and Computability	spring
CSCI 124: Operating Systems	fall
CSCI 130: Information Management	fall
CSCI 140: Software Engineering	fall
CSCI 150: Net-Centric Computing	spring
CSCI 160: Capstone - Applications of Computing to Other Disciplines	fall
CSCI 198: Topics in Computer Science	fall in even numbered years
CSCI 199: Independent Study in Computer Science	fall and spring

Table 3: Proposed offerings schedule.

Year	Fall	Spring
Freshman		CSCI 1, MATH 3
Sophomore	CSCI 2, CSCI 23	CSCI 100, CSCI 10
Junior	CSCI 124, CSCI 140	CSCI 25, CSCI 150
Senior	CSCI 130, CSCI 160	CSCI 101

Table 4: A possible student schedule.

4.6 Suggested Student Schedule

Table 4 presents one of several possible schedules a student might follow to complete the computer science major requirements.

5 Transition Plan for Current Majors and Minors

Because the first two courses in the proposed major were offered in 2006-2007, we do not anticipate a complicated transition for our students. We will provide information and guidelines, and we will meet with prospective and declared majors to develop and approve their programs. In no circumstances will we approve a major containing fewer credits from the department than either the existing or proposed majors requires.

Transition offerings: As of Fall 2008, we will offer the proposed curriculum on the proposed schedule. In 2007-2008, we will offer the proposed curriculum with only two changes:

In 2007-2008, we will offer these two courses from the existing curriculum:

- CSCI 105: Computability (fall)
- CSCI 101: Algorithms (spring)

In 2007-2008, we will not offer these two courses from the proposed curriculum:

- CSCI 101: Algorithm Analysis and Computability (spring)
- CSCI 130: Information Management (fall)

Guidance for students: The following paragraphs outline courses existing majors may take to fulfill major requirements under the proposed curriculum.

Although the title and course description of CSCI 101 will change, either version can be used towards the requirements of either version of the major.

Declared computer science majors who plan to graduate in 2008 or 2009 must complete the existing requirements with the following substitutions for required courses that will not be offered in 2007-2008 as expected:

- For CSCI 103: Computer Organization, take CSCI 100, 124, or 150
- For CSCI 171: Seminar, take CSCI 140 or 160
- For upper-level elective credit, take CSCI 10, 130, 140, 150, or 160 and request approval from the Director of Computer Science before beginning the course. Approval will be given to students who have not received credit for a course with the same content.

Students planning to graduate in May 2010 or later must fulfill the proposed requirements. Majors and minors may use these 2006-2007 courses in the new curriculum as follows:

- CSCI 107: Programming Languages replaces CSCI 1: Introduction to Computers and Computing
- CSCI 141: Databases replaces CSCI 130: Information Management
- Majors who declared a computer science major and completed MATH 8 (Calculus II) at Drew for that major before Fall 2007 may substitute MATH 8 for the MATH 3 requirement.

6 Computer Science Minor

Proposed requirements: We propose changing the requirements for the computer science minor to the following:

- CSCI 1: Introduction to Computers and Computing
- CSCI 2: Object-Oriented Programming
- CSCI 10: Human Interaction with Technology and Information
- CSCI 23: Discrete Mathematics for Computer Science or CSCI 25: Data Structures
- CSCI 100: Systems Programming and Tools
- One additional upper level course in Computer Science.

Specific changes: Individual changes required to transform the existing requirements into those proposed are as follows:

- Add CSCI 1: Introduction to Computers and Computing
- Add CSCI 10: Human Interaction with Technology and Information
- Add CSCI 100: Systems Programming and Tools
- Add the requirement that minors take one upper level course in Computer Science.
- Add the option of taking either CSCI 23: Discrete Mathematics for Computer Science or CSCI 25: Data Structures
- Remove CSCI 101: Algorithms
- Remove the requirement for 12 additional upper-level credits in computer science.

The total number of hours required to complete the computer science minor will be reduced from 28 to 24.

7 Joint Mathematics / Computer Science Major

We propose suspending admission to the joint Math/CS major until the department has completed its self-study which we anticipate will be in Spring 2007. During this process the department will review this major and propose changes we feel are appropriate given the new direction the proposed computer science curriculum represents. There is currently only one declared joint Math/CS major. This student will be accommodated using the transition plan described in Section 5.

References

[1] Drew university career center survey: career outcomes of the class of 2003, 2003.

[2] Drew university career center survey: career outcomes of the class of 2004, 2004.

[3] Drew university career center survey: career outcomes of the class of 2005, 2005.

[4] Carl Chang, Peter J. Denning, James H. Cross II, Gerald Engel, Robert Sloan, Doris Carver, Richard Eckhouse, Willis King, Francis Lau, Susan Mengel, Pradip Srimani, Eric Roberts, Russel Shackelford, Richard Austing, C. Fay Cover, Gordon Davies, Andrew McGettrick, G. Michael Schneider, and Ursula Wolz. Computing curricula 2001: Computer science. The Joint Task Force on Computing Curricula, IEEE Computer Society and the Association for Computing Machinery, December 2001.

[5] National Science Foundation. Characteristics of recent science and engineering graduates: 2003, 2006.

[6] Science planning committee report: Science at drew university: Taking it to the next level. Drew University, May 2006.

[7] Allen B. Tucker, Bruce H. Barnes, Robert M. Aiken, Keith Barker, Kim B. Bruce, J. Thomas Cain, Susan E. Conry, Gerald L. Engel, Richard G. Epstein, Doris K. Lidtke, Michael C. Mulder, Jean B. Rogers, Eugene H. Spafford, and A. Joe Turner. Computing curricula '91. Association for Computing Machinery and the Computer Society of the Institute of Electrical and Electronics Engineers, 1991.