CS Curriculum Change Request

The BYUH Computer Science department is undertaking a substantial update to its curriculum for Fall 2002 and beyond. The following explanations are offered in support and overview of the accompanying formal request documentation.

Contents

1 Overview

The Computer Science (CS) department is undertaking a substantial update to its curriculum. Most of the courses offered are changing from three credits to two credits, many of the old electives have become required, and a few new electives are being introduced. This change is motivated by three influences.

First and most important, the department feels a need to move toward **program accreditation** over the next five to seven years. This affects the core content. Over time new topics have gained prominence and must be included in the curriculum. At the same time, some older topics have faded and are no longer considered "core." This has caused some content areas to grow while others shrink, creating pressure to rebalance the division of material among the courses. This is especially timely since the major professional bodies in CS have just completed (December 2001) their ten-year update to the recommended CS curriculum.

Second, the department desires to capitalize on the **overlap of course work** between CS, Information Systems (IS), and Math in the new School of Computing, by combining near-redundant

courses. This also affects the division of course material among the courses.

Third, in the area of elective course work, the department desires to offer advanced courses in the areas of the **faculty research interests**. With a turn over in faculty recently, the elective course work needs to be refocused on the strengths and interests of the current faculty members.

1.1 Program Accreditation

Program accreditation serves several purposes. As part of our ongoing departmental review and improvement, we have determined that accreditation is a standard we should follow in our quest to become a "Center of Excellence" on the BYUH campus. The accreditation standards and the model curriculum standards are a measure against which we can compare ourselves as we look for ways to improve. In addition, accreditation helps us to establish and secure our academic reputation. Very small department size conveys a message of inadequacy to some observers, which message we must overcome by establishing a good reputation. Good reputation in turn improves our students' ability to get into graduate school, and our own ability to attract high-calibre students from our target areas, including Asia.

Computer Science accreditation is performed by ABET, formerly the Accreditation Board for Engineering and Technology. Their web site is at (abet.org). They handle program accreditation for both Computer Science (CS) and Information Systems (IS). We desire to comply with the accreditation requirements so that in time the BYUH CS department can become accredited.

In addition to the accreditation requirements, Computer Science has two major professional bodies that have over the past several decades developed standardized curricula that are very influential in CS departments around the world. These two organizations are the Computer Society of the Institute of Electrical and Electronic Engineers (IEEE-CS) and the Association for Computing Machinery (ACM). On December 15, 2001, after several years of development, they jointly released their ten-year curriculum update for CS. We believe its recommendations to be appropriate and we desire to comply.

1.2 Overlapping Course Work in CS and IS

Strictly speaking, there is no overlap in the course work currently taught by CS, IS, and Math. That is because the courses are directed at differing groups of students with different backgrounds and learning rates. However, the School of Computing has recognized that some of the courses have similar learning objectives and content, and might be consolidated if done carefully. We have pursued this opportunity.

A typical result would be a single course taught with greater frequency rather than two courses taught less frequently. This gives the students greater flexibility in planning their education. Another result would be the mixing of students from several majors in the same class. This gives the students a better sense of their own abilities by comparing with students who have different backgrounds. Such

mixing should improve understanding and reduce competitive fears as the students get to know each other better.

New students frequently ask, "What is the difference between CS and IS?" This is hard to explain since they lack the background to understand the answer. In introductory courses there is a further benefit that a student may decide to change majors early rather than finding out later that they should have changed majors. This is particularly true because CS and IS have much in common. It is true to a lesser degree between CS and Math.

The disadvantage of combining courses is that presentations become more general and do not cater to individual students as well as a "majors-only" class could do.

We think the advantages are worth the cost.

1.3 Faculty Research Interests

In the area of elective course work, the department desires to offer advanced courses in the areas of the faculty research interests. **The department feels it important that each tenured or tenure-track faculty member should teach one or more courses each year that are directly related to their historic or current research interests.** With a turn over in faculty recently, the elective course work needs to be refocused on the strengths and interests of the current faculty members.

Brother Colton did his PhD research in the area of automatic speech recognition by computers. Brother Fife is doing his PhD research in the area of mobile computing but has also done work in robotics and related areas. In each case we have created an elective course to focus on that area.

1.4 Creation of a CS Minor

We also wish to recognize and encourage those students who take several computer science classes in connection with another major. This can happen when a CS major decides to change to Math or IS or something else. It can also happen when a student majoring in Math or IS or something else has particular interest in CS. Currently there is no minor in CS for these students. By correcting that shortcoming, we will both recognize students for their CS course work, and we will encourage students who are close to a minor to take another class or two, thus increasing student knowledge and our class sizes with small or zero cost to the university.

2 FAQ: a Few Asked Questions

Following are a few questions that have been asked about this proposal.

2.1 Summer Term Course Offerings

Question: As many students are here during the summer (sponsered etc) how do you cater for their needs with classes during the Summer. They can get out quicker if they can take classes during the Summer as well as the Spring.

Answer: A larger portion of our students can take classes in Spring. Since we must choose between committing a faculty member to Spring or Summer, and since we are short-handed, we have chosen to commit our Spring/Summer teaching time to Spring. For students who wish to attend Summer, we recommend that they take one or more General Education courses.

We do not expect to offer any classes during Summer in the foreseeable future.

2.2 Math 110 and 111

Question: As you require Math 119, and this course has prerequisites of Math 111 and Math 110 or competency, how do you account for those students that have to take these two classes??

Answer: Most of our students can take Math 119 when they arrive. We do not consider Math 110 or Math 111 to be a hidden prerequisite since it only affects a minority of our students.

3 Changing Graduation Requirements

To comply with accreditation requirements (ABET), standard curricula (CC2001), and our own professional sense of what should be required, we request a change to the graduation requirements for the CS degree. The new standard will be more demanding than our current performance.

ABET is the Accrediting Body for Engineering and Technology. It is the body that accredits CS programs. Formerly such programs were accredited by CSAB, the Computer Science Accreditation Board, but CSAB merged with ABET recently. The current ABET standards for Computer Science were issued in November of 2001.

CC2001 is the Computing Curricula 2001 standard created and approved by IEEE-CS and ACM, the two major professional bodies in Computer Science. CC2001 was issued on December 15, 2001. The prior revision was in 1991, and was widely adopted by CS programs throughout the world.

3.1 Accreditation Requirements

Accreditation requirements form the baseline that we would like to achieve. They are:

CS: 40 hours, including 24 hours of introductory and intermediate material and 16 hours of advanced material. 32 hours are required topics, marked "R" in the charts that follow, and 8 hours

are available for advanced elective course work.

Math/Science: 30 hours as follows: **Math: 15–18 hours,** including Calculus (Math 112 and 113 or Math 119), and Statistics (Math 221 or 321), and Discrete Mathematics (Math 201/L and Math 202/L). **Science: 12–15 hours,** including a two-semester lab sequence (either Physics 121 and 122, 6 hours, or Chemistry 105, 105L, 106, and 106L, 8 hours) and Biology (3 hours) general education class.

Arts and Humanities: 30 hours, including humanities, social sciences, arts, and other disciplines that serve to broaden the background of the student. Courses in CS, IS, mathematics, and science can not be counted in this total. Students will naturally earn 37 credits or more by completing BYUH General Education and Religion requirements.

The "40 CS, 30 Math/Sci, 30 Arts" requirement is specified by ABET. The "24 intro, 16 advanced" requirement is specified by CC2001. Core knowledge units are specified loosely by ABET and precisely by CC2001.

The 40/30/30 hour requirement can be restated by netting out BYUH requirements (aka double dipping). We can identify 2/10/30 credits that satisfy both accreditation and BYUH standards, leaving the CS major with a burden of 38/20/0 or 40/20/0 (58 or 60 major hours).

3.2 BYUH Requirements

BYUH has a campus-wide standard calling for students to graduate with 120 credit hours in four years, but still allowing substantial free elective course work. To facilitate this we have cut back from strict accreditation requirements as follows.

CS: 40 hours as written above.

Math/Science: 20 hours (down 10) as follows: **Math: 11 hours,** including Calculus (Math 119), and Statistics (Math 221), and Discrete Mathematics (Math 201 and 202; 201L and 202L count as CS). **Science: 9 hours,** including a two-semester lab sequence (Physics 121 and 122, 6 hours, or at the student's option, Chemistry 105, 105L, 106, and 106L, 8 hours) and Biology (3 hours) general education class.

Arts and Humanities: 30 hours as written above.

The revised 40/20/30 hour total, net of 0/10/30 BYUH requirements, leaves the CS major with a burden of 40/10/0, or 50 major hours, which we believe meets BYUH guidelines.

3.3 Side-By-Side Credit-Hour Accounting of Graduation Requirements

This table presents the current graduation requirements (2001-2003 catalog) and the proposed graduation requirements.

Current		Proposed		Dif
Computer Science Major Requirements				
CS 142	3	CS 101/201	5	+2
CS 143	3	CS 210	2	-1
CS 235	3	MATH 201/L	3	
CS 236	3	MATH 202/L	3	
CS 240	3	CS 202	3	
CS 252	3	CS 320	2	-1
CS 312	3	CS 301	3	
CS 330	3	CS 420	2	-1
CS 345	3	CS 415	2	-1
CS 380	3	CS 410	2	-1
(new)		CS 333	2	+2
(new)		CS 433	2	+2
(new)		CS 440	2	+2
(new)		IS 280	2	+2
(new)		IS 351	3	+3
CS ele	9	CS ele	6	-1
PHYS 122	3	PHYS 122	3	
MATH 221	3	MATH 221	3	
MATH 343	3	(delete)		-3
Total CS	48	Total CS	50	+2

Current		Proposed		Dif
General Education Requirements				
MATH 119	4	MATH 119	4	
Biology	3	Biology	3	
PHYS 121	3	PHYS 121	3	
other	25	other	25	
Total GE	35	Total GE	35	0

Summary of Graduation Requirements				
Total CS	48	Total CS	50	+2
Total GE	35	Total GE	35	
Religion	14	Religion	14	
Electives	23	Electives	21	-2
Total	120	Total	120	0

Within the normal process of a BYUH education, students will take at least 23 hours of non-science general education and 14 hours of religion that together filfull the CS major requirement for 30 hours in arts and humanities. Also a Biology class will be taken, helping to satisfy the CS major requirement for Science.

Additionally, Mathematics 119 and Physics 121 are required. This can be considered "double dipping" because they satisfy both major requirements (dictated by accreditation) and BYUH general education requirements, but students could have satisfied the general education requirement in some other way that did not satisfy major requirements.

Finally, students can "double dip" again, at their option, by completing of CS 491, 492, 493 (Seminar), which counts for both CS elective credit (3 hours) and for General Education Area III Advanced Writing, replacing English 301 (2 credits). Many students take this option.

3.4 Proposed Academic Advisor Checklist

This is a sample Academic Advisor Checklist for new CS students.

Course #	Title	Hrs	Prerequisites	Offered	Old #
MAJOR REQUIREMENTS: *complete all*					
CS 101	Computer Programming I	2	Math 100	F W S	IS 230
CS 201	Computer Programming II	3	CS 101, Math 110	F W S	IS 231
CS 202	Computer Programming III	3	CS 201, Math 201/L	F W	CS 240
CS 210	Computer Organization	2	CS 101, Math 110	W	CS 143
CS 301	Algorithms & Complexity	3	CS 202, Math 202/L	F W	CS 312
CS 320	Computational Theory	2	CS 201, Math 201/L	F	CS 252
CS 333	Software Engineering I	2	CS 202	F	CS 428
CS 410	Computer Architecture	2	CS 210, CS 301	F	CS 380
CS 415	Operating Systems Design	2	CS 210, CS 301	F	CS 345
CS 420	Programming Languages	2	CS 202, CS 320	W	CS 330
CS 433	Software Engineering II	2	CS 333	W	CS 428
CS 440	Intelligent Systems	2	CS 301	W	CS 470

SUPPORTING COURSES: *complete all*					
IS 280	Data Communications	2	CS 101	F W S	CS 460
IS 351	Adv Data Files / Databases	3	IS 350 or CS 202	F W S	CS 452
Math 119	Intro to Calculus	4	Math 110, 111	F	same
or 112	Calculus I	5	Math 111	F W	same
Math 201/L	Discrete Mathematics I	3	Math 110	F W S	CS 235
Math 202/L	Discrete Mathematics II	3	Math 201/L	F W	CS 236
Math 221	Principles of Statistics I	3	Math 110	F W S	same
or 321	Mathematical Statistics	3	Math 214 or consent		same
Phys 121	General Physics I	3	Math 119	F	same
Phys 122	General Physics II	3	Phys 121	W	same

ELECTIVES: *select 8 credit hours*					
CS 421	Algorith Lang & Compilers	2	CS 420	var	CS 431
CS 441	Auto Speech Recognition	2	CS 440	var	new
CS 442	Image Proc & Comp Vision	2	CS 440	var	new
CS 451	Advanced Database Topics	2	CS 301, IS 351	var	CS 453
CS 456	Mobile Computing	2	CS 301, IS 351	var	new
CS 461	Computer Graphics	2	CS 301, Math 343	var	CS 455
CS 491	Seminar I	1	consent	var	same
CS 492	Seminar II	1	consent	var	same
CS 493	Seminar III	1	consent	var	same
IS 431	E-Commerce Web Development	3	consent	F W	same

3.5 Translating Forward: Old Courses Into New Numbers

This table presents the mapping between old and new course numbers in **old-number** order.

For simplicity in meeting graduation requirements, an old course of any type will be accepted in lieu of the new course of the same type and vice versa. "v" indicates variable credit, usually 1–3. "R" indicates a required class.

was	cr	R	short title	becomes	cr	R	short title
CS 142	3	R	Intro Computer Progmg	CS 101	2	R	Computer Programming I
			(becomes two courses)	CS 201	3	R	Computer Programming II
CS 143	3	R	Computer Organization	CS 210	2	R	Computer Organization
CS 220	3		Digital Logic Design				**deleted**
CS 235	3	R	Foundations of CS I	M 201/L	3	R	Discrete Mathematics I
CS 236	3	R	Foundations of CS II	M 202/L	3	R	Discrete Mathematics II
CS 240	3	R	Adv Programming Conc	CS 202	3	R	Computer Programming III
CS 252	3	R	Computational Theory	CS 320	2	R	Computational Theory
CS 312	3	R	Algorithm Analysis	CS 301	3	R	Algorithms & Complexity
CS 330	3	R	Conc of Pgmg Languages	CS 420	2	R	Programming Languages
CS 345	3	R	Operating Sys Design	CS 415	2	R	Operating Systems Design
CS 380	3	R	Computer Architecture	CS 410	2	R	Computer Architecture
CS 390R	v		Special Topics in CS	CS 390R	v		Special Topics in CS
CS 399R	v		Coop Education in CS	CS 399R	v		Coop Education in CS
CS 428	3		Software Sys Des & Impl	CS 333	2	R	Software Engineering I
			(becomes two courses)	CS 433	2	R	Software Engineering II
CS 429	3		Software Testing				**deleted**
CS 431	3		Algor Lang & Compilers	CS 421	2		Algorith Lang & Compilers
CS 452	3		Database Modeling Conc	IS 351	3	R	Adv Data Files / Databases
CS 453	3		Adv Data Structures	CS 451	2		Advanced Database Topics
CS 455	3		Computer Graphics	CS 461	2		Computer Graphics
CS 460	3		Communic & Networking	IS 280	2	R	Data Communications
CS 470	3		Artificial Intelligence	CS 440	2	R	Intelligent Systems
CS 491	1		Seminar I	CS 491	1		Seminar I
CS 492	1		Seminar II	CS 492	1		Seminar II
CS 493	1		Seminar III	CS 493	1		Seminar III
CS 495R	v		Independent Study	CS 495R	v		Independent Study in CS
CS 496R	v		Student Research	CS 496R	v		Student Research in CS
			new	CS 456	2		Mobile Computing
			new	CS 441	2		Auto Speech Recognition
			new	CS 442	2		Image Proc & Comp Vision
			new	CS 490R	2		Advanced Topics in CS

3.6 Translating Backward: New Courses Into Old Numbers

This table presents the mapping between old and new course numbers in **new-number** order.

For simplicity in meeting graduation requirements, an old course of any type will be accepted in lieu of the new course of the same type and vice versa. "v" indicates variable credit, usually 1–3. "R" indicates a required class.

was	cr	R	short title	**becomes**	cr	R	short title
CS 142	3	R	Intro Computer Progmg	**CS 101**	2	R	Computer Programming I
			(becomes two courses)	**CS 201**	3	R	Computer Programming II
CS 240	3	R	Adv Programming Conc	**CS 202**	3	R	Computer Programming III
CS 143	3	R	Computer Organization	**CS 210**	2	R	Computer Organization
CS 312	3	R	Algorithm Analysis	**CS 301**	3	R	Algorithms & Complexity
CS 252	3	R	Computational Theory	**CS 320**	2	R	Computational Theory
CS 428	3		Software Sys Des & Impl	**CS 333**	2	R	Software Engineering I
CS 390R	v		Special Topics in CS	**CS 390R**	v		Special Topics in CS
CS 399R	v		Coop Education in CS	**CS 399R**	v		Coop Education in CS
CS 380	3	R	Computer Architecture	**CS 410**	2	R	Computer Architecture
CS 345	3	R	Operating Sys Design	**CS 415**	2	R	Operating Systems Design
CS 330	3	R	Conc of Pgmg Languages	**CS 420**	2	R	Programming Languages
CS 431	3		Algor Lang & Compilers	**CS 421**	2		Algorith Lang & Compilers
			(see CS 428)	**CS 433**	2	R	Software Engineering II
CS 470	3		Artificial Intelligence	**CS 440**	2	R	Intelligent Systems
			new	**CS 441**	2		Auto Speech Recognition
			new	**CS 442**	2		Image Proc & Comp Vision
CS 453	3		Adv Data Structures	**CS 451**	2		Advanced Database Topics
			new	**CS 456**	2		Mobile Computing
CS 455	3		Computer Graphics	**CS 461**	2		Computer Graphics
			new	**CS 490R**	2		Advanced Topics in CS
CS 491	1		Seminar I	**CS 491**	1		Seminar I
CS 492	1		Seminar II	**CS 492**	1		Seminar II
CS 493	1		Seminar III	**CS 493**	1		Seminar III
CS 495R	v		Independent Study	**CS 495R**	v		Independent Study in CS
CS 496R	v		Student Research	**CS 496R**	v		Student Research in CS
CS 460	3		Communic & Networking	**IS 280**	2	R	Data Communications
CS 452	3		Database Modeling Conc	**IS 351**	3	R	Adv Data Files / Databases
CS 235	3	R	Foundations of CS I	**M 201/L**	3	R	Discrete Mathematics I
CS 236	3	R	Foundations of CS II	**M 202/L**	3	R	Discrete Mathematics II

100- and 200-level courses (CS, IS, or Math) are introductory. 300-level courses are intermediate. 400-level courses are advanced.

3.7 Graduation Requirements: Catalog Copy

Computer Science (B.S.)

Students are required to complete (a) 40 credits of Computer Science, with at least 16 at the 400 level, (b) 30 credits of mathematics and science, and (c) 30 credits of arts and humanities (general education and religion satisfy this).

The mathematics and science area must include 15 or more credits of Mathematics (including statistics, discrete mathematics, and calculus) and 12 or more credits of Science (including a two-semester sequence or equivalent in a laboratory science). The six-credit requirement for General Education Area II Introduction to the Natural World helps fulfill this requirement.

Required Computer Science Courses		**(34 credits)**
CS 101	Computer Programming I	(2)
CS 201	Computer Programming II	(3)
CS 202	Computer Programming III	(3)
CS 210	Computer Organization	(2)
CS 301	Algorithms & Complexity	(3)
CS 320	Computational Theory	(2)
CS 333	Software Engineering I	(2)
CS 410	Computer Architecture	(2)
CS 415	Operating Systems Design	(2)
CS 420	Programming Languages	(2)
CS 433	Software Engineering II	(2)
CS 440	Intelligent Systems	(2)
IS 280	Data Communications	(2)
IS 351	Adv Data Files / Databases	(3)
Math 201L	Discrete Math I Lab	(1)
Math 202L	Discrete Math II Lab	(1)

Elective Computer Science Courses		**(select 6* credits)**
CS 421	Compilers	(2)
CS 441	Auto Speech Recognition	(2)
CS 442	Image Processing and Computer Vision	(2)
CS 451	Advanced Database Topics	(2)
CS 456	Mobile Computing	(2)
CS 461	Computer Graphics	(2)
CS 490R	Advanced Topics in Computer Science	(var)
CS 491 *	Seminar I	(1)
CS 492	Seminar II	(1)
CS 493 *	Seminar III	(1)
CS 495R	Independent Study in Computer Science	(var)
CS 496R	Student Research in Computer Science	(var)
IS 431	E-Commerce Web Development	(3)

* Completion of CS 491 and 493 also fulfills the two-credit requirement for General Education Area III Advanced Writing, effectively reducing the Elective requirement to 6 additional credits, one of which must then be CS 492.

Mathematics and Science Courses

Required Mathematics Courses		**(11* credits)**
Math 119 *	Introduction to Calculus	(4)
or 112	Calculus I	(5)
Math 201	Discrete Math I	(2)
Math 202	Discrete Math II	(2)
Math 221	Principles of Statistics	(3)

* This fulfills the General Education Area I Mathematics Track requirement, effectively reducing the Math requirement to 7 additional credits.

Required Science: Physics Option		**(9 credits)**
Biology	General Education requirement	(3)
Phys 121	General Physics	(3)
Phys 122	General Physics	(3)

Required Science: Chemistry Option		**(11 credits)**
Biology	General Education requirement	(3)
Chem 105/L	General Chemistry I, Lab	(4)
Chem 106/L	General Chemistry II, Lab	(4)

* The six-credit requirement for General Education Area II Introduction to the Natural World helps fulfill this requirement, effectively reducing the science requirement to 3 additional credits.

Recommended Mathematics and Science Courses **(10 credits)**

To prepare for graduate school, students are urged to take a minimum of 15 to 18 credits of Mathematics, and 12 to 15 credits of Science, for a total of 30 credits of Mathematics and Science. Note that by careful scheduling, a student may be able to earn a minor in Mathematics while preparing for graduate school.

Summary of Degree Requirements

Computer Science Core	34 credits
Computer Science Elective	6 credits
Support Math Courses (except Gen Ed)	7 credits
Support Science Courses (except Gen Ed)	3 credits
Total CS Program	**50 credits**
Area I (using Math 119)	9 credits
Area II (using Physics 121)	15 credits
Area III	11 credits
General Education	**35 credits**
Religion	**14 credits**
Electives	**21 credits**
Total Requirements	**120 credits**

Sample 15-15 Schedule for a Four-Year CS Student

Year	Fall	cr	Winter	cr
1	CS 101	2	CS 201	3
	BIOL 100	3	CS 210	2
	ART ele	3	IS 280	2
	elective	1	MATH 201/L	3
	ENGL 101	3	ENGL 201	3
	REL 121	2	REL 122	2
	EXS 177	1		
		15		15
two-year mission				
2	CS 202	3	CS 301	3
	CS 320	2	CS 420	2
	MATH 202/L	3	IS 351	3
	MATH 119	4	MATH 221	3
	REL 211	2	REL 212	2
	EXS ele	1	elective	2
		15		15
3	CS 491	1	CS 492	1
	CS 333	2	CS 433	2
	CS 410	2	CS 440	2
	CS 415	2	elective	2
	PHYS 121	3	PHYS 122	3
	HIST 201	3	HIST 202	3
	REL 324	2	REL 325	2
		15		15
4	CS 493	1	CS ele	2
	CS ele	2	CS ele	2
	MATH ele	3	MATH ele	3
	SCI ele	3	IDS ele	3
	REL ele	2	PSYC 111	3
	elective	4	elective	2
		15		15

Sample 12-12-6 Schedule for a Four-Year CS Student

Year	Fall	cr	Winter	cr	Spring	cr
1	CS 101	2	CS 201	3	BIOL 100	3
	ENGL 101	3	CS 210	2	ENGL 201	3
	ART ele	3	IS 280	2		
	elective	1	MATH 201/L	3		
	REL 121	2	REL 122	2		
	EXS 177	1				
		12		12		6
2	CS 202	3	CS 301	3	MATH 221	3
	CS 320	2	CS 420	2	REL 211	2
	MATH 202/L	3	IS 351	3	EXS ele	1
	MATH 119	4	REL 212	2		
			elective	2		
		12		12		6
3	CS 491	1	CS 492	1	HIST 201	3
	CS 333	2	CS 433	2	REL 325	2
	CS 410	2	CS 440	2	elective	1
	CS 415	2	elective	1		
	PHYS 121	3	PHYS 122	3		
	REL 324	2	HIST 202	3		
		12		12		6
4	CS 493	1	CS ele	2	PSYC 111	3
	CS ele	2	CS ele	2	SCI ele	3
	MATH ele	3	MATH ele	3		
	REL ele	2	IDS ele	3		
	elective	4	elective	2		
		12		12		6

Sample 15-15-6 Schedule for a Two-Year Transfer CS Student

Year	Fall	cr	Winter	cr	Spring	cr
1	CS 201	3	CS 202	3	IS 351	3
	IS 280	2	CS 210	2	MATH 221	3
	MATH 201/L	3	MATH 202/L	3		
	MATH 112	5	MATH 113	5		
	REL 121	2	REL 122	2		
		15		15		6
2	CS 301	3	CS 440	2		
	CS 320	2	CS 420	2		
	CS 333	2	CS 433	2		
	PHYS 121	3	PHYS 122	3		
		10		9		
2	CS 410	2				
	CS 415	2				
	CS ele	2	CS ele	2		
	CS ele	2	CS ele	2		
		8		4		

Computer Science (minor)

A minor in Computer Science prepares a student for intensive computer involvement within another discipline, such as science, business, or the arts.

Required Courses **(14 credits)**

CS 101	Computer Programming I	(2)
CS 201	Computer Programming II	(3)
CS 202	Computer Programming III	(3)
Math 201/L	Discrete Math I	(3)
Math 202/L	Discrete Math II	(3)

4 Joint CS/IS Introductory Sequence

The School of Computing has recognized that the CS and IS introductory courses have similar learning objectives and content, and might be consolidated if this is done carefully.

The recently-released internationally-recognized CS curriculum requires programming proficiency and study of discrete mathematics. The anticipated internationally-recognized IS curriculum also requires programming proficiency and study of discrete mathematics, but not at the same level as CS. Statistics (Math 221 or Math 321) remains unchanged as a requirement for both CS and IS.

4.1 The New CS Introductory Sequence

The new CS introductory sequence consists of three courses in programming and two courses in discrete mathematics.

was	cr	R	short title	becomes	cr	R	short title
CS 142	3	R	Intro Computer Progmg	CS 101	2	R	Computer Programming I
			(becomes two courses)	CS 201	3	R	Computer Programming II
CS 240	3	R	Adv Programming Conc	CS 202	3	R	Computer Programming III
CS 235	3	R	Foundations of CS I	Math 201/L	3	R	Discrete Mathematics I
CS 236	3	R	Foundations of CS II	Math 202/L	3	R	Discrete Mathematics II

4.2 The Future IS Introductory Sequence

The planned new IS introductory sequence consists of two courses in programming and one course in discrete mathematics. This change in IS graduation requirements will be submitted separately at a future time, but for context it seems useful to mention it here.

was	cr	R	short title	becomes	cr	R	short title
IS 230	3	R	Computer Programming I	CS 101	2	R	Computer Programming I
IS 231			Computer Programming II	CS 201	3	R	Computer Programming II
			new	Math 201/L	3	R	Discrete Mathematics I

5 Introductory Core (Required) Courses

5.1 CS 101: Computer Programming I (2)(F,W,SP)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Computer Pgmg 1`

CS 101	**Computer Programming I (2)(F,W,SP)**
new	Structured programming fundamentals: control (sequence,
required	selection, iteration, and subroutine), data type (int, float, array), and output formatting. Extensive program development and testing. (Prerequisite: Math 100.)

This is derived from, and replaces both:

IS 230	**Computer Programming I (3)(F,W,SP)**
old	Structured programming fundamentals: Control (sequence,
IS req	selection, iteration, and subroutine), data type (int, float, and output formatting). Extensive program development and testing.

CS 142	**Introduction to Computer Programming (3)(F)**
old	Introduction to program design and development. Principles of
CS req	algorithm formation and implementation. (Prerequisite: Algebra.)

CC2001 Knowledge Units: CS 101 will cover the following CC2001 knowledge units: PF1 (fundamental programming constructs, 9h), SP1 (history of computing, 1h).

CS 101 meets three times a week for one hour each time. One hour a week is devoted to lecture. The other two hours are devoted to in-class lab, with the instructor present. Experience with IS 230 shows that about half of the students can complete all of their work during this in-class lab time. Out-of-class lab is also offered most evenings from 6 PM to midnight in GCB 101. Experience with IS 230 shows that about half of the students required out-of-class support, with some spending nine or more hours per week (in class and out) to complete the assignments.

CS 101 (Programming 1) will directly replace IS 230. It will be taken by all CS, IS, and Math Ed students. CS and IS students would typically take it the first semester of their freshman year. Advanced students would be permitted to test out of the class, either for credit or not. We assume 80% of the students have never programmed before. (This matches our historic experience in IS 230 and CS 142.) The class is taught using the C programming language. Completion of CS 101 constitutes "programming proficiency" for a Math Ed major. We anticipate seven sections per year with an average enrollment of 30 students per section.

CS 101 Syllabus (Relevant CC2001 Excerpts)

PF1. Fundamental programming constructs [core]

Minimum core coverage time: 9 hours

Topics:
- Basic syntax and semantics of a higher-level language
- Variables, types, expressions, and assignment
- Simple I/O
- Conditional and iterative control structures
- Functions and parameter passing
- Structured decomposition

Learning objectives:

1. Analyze and explain the behavior of simple programs involving the fundamental programming constructs covered by this unit.

2. Modify and expand short programs that use standard conditional and iterative control structures and functions.

3. Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.

4. Choose appropriate conditional and iteration constructs for a given programming task.

5. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces.

6. Describe the mechanics of parameter passing.

SP1. History of computing [core]

Minimum core coverage time: 1 hour

Topics:
- Prehistory-the world before 1946
- History of computer hardware, software, networking
- Pioneers of computing

Learning objectives:

1. List the contributions of several pioneers in the computing field.

2. Compare daily life before and after the advent of personal computers and the Internet.

3. Identify significant continuing trends in the history of the computing field.

5.2 CS 201: Computer Programming II (3)(F,W,SP)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Computer Pgmg 2`

CS 201	**Computer Programming II (3)(F,W,SP)**
new	Review of CS 101. Problem solving, stacks, queues, hash tables,
required	mathematical analysis of algorithms, regular expressions. Web programming including CGI and database. (Prerequisites: CS 101 and Math 110; or Math 112; or Math 119.)

This is derived from, and replaces:

IS 231	**Computer Programming II (3)(F,W,SP)**
old	Basic control structures plus recursion. Data structures
elective	including strings, arrays, records, pointers, lists, and trees. Extensive program documentation, design, development, and testing. (Prerequisite: IS 230.)

CC2001 Knowledge Units: CS 201 will cover the following CC2001 knowledge units: PF1 (fundamental programming constructs, 9h), PF2 (algorithms and problem solving, 6h), PF4 (recursion, 5h), AL1 (basic algorithmic analysis, 4h), NC4 (the web as an example of client-server computing, 3h), IM5 (database query languages, non-core).

IS 231 is taken by about 25% of IS students and by some CS students. The class is taught using the Perl programming language. It is taught in three sections per year with a typical enrollment of 20 students each.

CS 201 (Programming 2) will directly replace IS 231. It will be taken by all CS, IS, and Math (not Ed) students. CS students will take it the second semester of their freshman year. IS and Math students may take it later. The prerequisite structure allows Math majors to take CS 201 directly without taking CS 101, but only after demonstrating maturity by completing a Calculus class. We believe with that level of maturity the students would be able to come up to speed quickly. Completion of CS 201 would constitute "programming proficiency" for an IS major or a Math major. We anticipate seven sections per year with an average enrollment of 30 students per section.

CS 201 Syllabus (Relevant CC2001 Excerpts)

PF1. Fundamental programming constructs [core]

Minimum core coverage time: 9 hours

Topics:
- Basic syntax and semantics of a higher-level language
- Variables, types, expressions, and assignment
- Simple I/O
- Conditional and iterative control structures
- Functions and parameter passing
- Structured decomposition

Learning objectives:

1. Analyze and explain the behavior of simple programs involving the fundamental programming constructs covered by this unit.

2. Modify and expand short programs that use standard conditional and iterative control structures and functions.

3. Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.

4. Choose appropriate conditional and iteration constructs for a given programming task.

5. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces.

6. Describe the mechanics of parameter passing.

PF2. Algorithms and problem-solving [core]

Minimum core coverage time: 6 hours

Topics:
- Problem-solving strategies
- The role of algorithms in the problem-solving process
- Implementation strategies for algorithms
- Debugging strategies
- The concept and properties of algorithms

Learning objectives:

1. Discuss the importance of algorithms in the problem-solving process.

2. Identify the necessary properties of good algorithms.

3. Create algorithms for solving simple problems.

4. Use pseudocode or a programming language to implement, test, and debug algorithms for solving simple problems.

5. Describe strategies that are useful in debugging.

PF4. Recursion [core]

Minimum core coverage time: 5 hours

Topics:
- The concept of recursion
- Recursive mathematical functions
- Simple recursive procedures
- Divide-and-conquer strategies
- Recursive backtracking
- Implementation of recursion

Learning objectives:

1. Describe the concept of recursion and give examples of its use.

2. Identify the base case and the general case of a recursively defined problem.

3. Compare iterative and recursive solutions for elementary problems such as factorial.

4. Describe the divide-and-conquer approach.

5. Implement, test, and debug simple recursive functions and procedures.

6. Describe how recursion can be implemented using a stack.

7. Discuss problems for which backtracking is an appropriate solution.

8. Determine when a recursive solution is appropriate for a problem.

AL1. Basic algorithmic analysis [core]

Minimum core coverage time: 4 hours

Topics:
- Asymptotic analysis of upper and average complexity bounds
- Identifying differences among best, average, and worst case behaviors
- Big O, little o, omega, and theta notation
- Standard complexity classes
- Empirical measurements of performance
- Time and space tradeoffs in algorithms
- Using recurrence relations to analyze recursive algorithms

Learning objectives:

1. Explain the use of big O, omega, and theta notation to describe the amount of work done by an algorithm.

2. Use big O, omega, and theta notation to give asymptotic upper, lower, and tight bounds on time and space complexity of algorithms.

3. Determine the time and space complexity of simple algorithms.

4. Deduce recurrence relations that describe the time complexity of recursively defined algorithms.

5. Solve elementary recurrence relations.

NC4. The web as an example of client-server computing [core]

Minimum core coverage time: 3 hours

Topics:
- Web technologies
  - Server-side programs
  - Common gateway interface (CGI) programs
  - Client-side scripts
  - The applet concept
- Characteristics of web servers
  - Handling permissions
  - File management
  - Capabilities of common server architectures
- Role of client computers
- Nature of the client-server relationship
- Web protocols
- Support tools for web site creation and web management
- Developing Internet information servers
- Publishing information and applications

Learning objectives:

1. Explain the different roles and responsibilities of clients and servers for a range of possible applications.

2. Select a range of tools that will ensure an efficient approach to implementing various client-server possibilities.

3. Design and build a simple interactive web-based application (e.g., a simple web form that collects information from the client and stores it in a file on the server).

IM5. Database query languages [elective]

Topics:

- Overview of database languages
- SQL (data definition, query formulation, update sublanguage, constraints, integrity)
- Query optimization
- QBE and 4th-generation environments
- Embedding non-procedural queries in a procedural language
- Introduction to Object Query Language

Learning objectives:

1. Create a relational database schema in SQL that incorporates key, entity integrity, and referential integrity constraints.

2. Demonstrate data definition in SQL and retrieving information from a database using the SQL SELECT statement.

3. Evaluate a set of query processing strategies and select the optimal strategy.

4. Create a non-procedural query by filling in templates of relations to construct an example of the desired query result.

5. Embed object-oriented queries into a stand-alone language such as C++ or Java (e.g., SELECT Col.Method() FROM Object).

5.3 CS 202: Computer Programming III (3)(F,W)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Computer Pgmg 3`

CS 202	**Computer Programming III (3)(F,W)**
new	Advanced software development with an object-oriented focus.
required	Development and testing of several 1500- to 2000-line modules from formal specifications. (Prerequisite: CS 201 and Math 201/L.)

This is derived from, and replaces:

CS 240	**Advanced Programming Concepts (3)(W)**
old	Advanced software development with an object-oriented focus.
required	Development and testing of several 1500- to 2000-line modules from formal specifications. Unix and C++ environment. (Prerequisite: CS 236.)

CC2001 Knowledge Units: CS 202 will cover the following CC2001 knowledge units: PF3 (fundamental data structures, 14h), PF4 (recursion, 5h), PF5 (event-driven programming, 4h), SP4 (professional and ethical responsibilities, 3h), SP5 (risks and liabilities of computer-based systems, 2h), SP6 (intellectual property, 3h), SP7 (privacy and civil liberties, 2h).

CS 240 is taken by all CS students, typically in their third semester. Typically 236 and 240 are offered at the same time, and students take them concurrently. This is a project-oriented course where students consolidate their experiences to date, and develop confidence to embark on more challenging projects in the future.

CS 202 (Programming 3) will replace CS 240. It will be taken by all CS students, typically their third semester. IS and Math students desiring a CS minor would also take this class. Completion of CS 202 would constitute "programming proficiency" for a CS major. We anticipate two sections per year with an average enrollment of 20 students per section.

CS 202 Syllabus (Relevant CC2001 Excerpts)

PF3. Fundamental data structures [core]

Minimum core coverage time: 14 hours

Topics:
- Primitive types
- Arrays
- Records
- Strings and string processing
- Data representation in memory
- Static, stack, and heap allocation
- Runtime storage management
- Pointers and references
- Linked structures
- Implementation strategies for stacks, queues, and hash tables
- Implementation strategies for graphs and trees
- Strategies for choosing the right data structure

Learning objectives:

1. Discuss the representation and use of primitive data types and built-in data structures.

2. Describe how the data structures in the topic list are allocated and used in memory.

3. Describe common applications for each data structure in the topic list.

4. Implement the user-defined data structures in a high-level language.

5. Compare alternative implementations of data structures with respect to performance.

6. Write programs that use each of the following data structures: arrays, records, strings, linked lists, stacks, queues, and hash tables.

7. Compare and contrast the costs and benefits of dynamic and static data structure implementations.

8. Choose the appropriate data structure for modeling a given problem.

PF4. Recursion [core]

Minimum core coverage time: 5 hours

Topics:
- The concept of recursion
- Recursive mathematical functions
- Simple recursive procedures
- Divide-and-conquer strategies

- Recursive backtracking
- Implementation of recursion

Learning objectives:

1. Describe the concept of recursion and give examples of its use.
2. Identify the base case and the general case of a recursively defined problem.
3. Compare iterative and recursive solutions for elementary problems such as factorial.
4. Describe the divide-and-conquer approach.
5. Implement, test, and debug simple recursive functions and procedures.
6. Describe how recursion can be implemented using a stack.
7. Discuss problems for which backtracking is an appropriate solution.
8. Determine when a recursive solution is appropriate for a problem.

PF5. Event-driven programming [core]

Minimum core coverage time: 4 hours

Topics:
- Event-handling methods
- Event propagation
- Exception handling

Learning objectives:

1. Explain the difference between event-driven programming and command-line programming.
2. Design, code, test, and debug simple event-driven programs that respond to user events.
3. Develop code that responds to exception conditions raised during execution.

SP4. Professional and ethical responsibilities [core]

Minimum core coverage time: 3 hours

Topics:
- Community values and the laws by which we live
- The nature of professionalism
- Various forms of professional credentialing and the advantages and disadvantages
- The role of the professional in public policy
- Maintaining awareness of consequences
- Ethical dissent and whistle-blowing
- Codes of ethics, conduct, and practice (IEEE, ACM, SE, AITP, and so forth)
- Dealing with harassment and discrimination

- "Acceptable use" policies for computing in the workplace

Learning objectives:

1. Identify progressive stages in a whistle-blowing incident.

2. Specify the strengths and weaknesses of relevant professional codes as expressions of professionalism and guides to decision-making.

3. Identify ethical issues that arise in software development and determine how to address them technically and ethically.

4. Develop a computer use policy with enforcement measures.

5. Analyze a global computing issue, observing the role of professionals and government officials in managing the problem.

6. Evaluate the professional codes of ethics from the ACM, the IEEE Computer Society, and other organizations.

SP5. Risks and liabilities of computer-based systems [core]

Minimum core coverage time: 2 hours

Topics:
- Historical examples of software risks (such as the Therac-25 case)
- Implications of software complexity
- Risk assessment and management

Learning objectives:

1. Explain the limitations of testing as a means to ensure correctness.

2. Describe the differences between correctness, reliability, and safety.

3. Discuss the potential for hidden problems in reuse of existing components.

4. Describe current approaches to managing risk, and characterize the strengths and shortcomings of each.

SP6. Intellectual property [core]

Minimum core coverage time: 3 hours

Topics:
- Foundations of intellectual property
- Copyrights, patents, and trade secrets
- Software piracy
- Software patents
- Transnational issues concerning intellectual property

Learning objectives:

1. Distinguish among patent, copyright, and trade secret protection.

2. Discuss the legal background of copyright in national and international law.

3. Explain how patent and copyright laws may vary internationally.

4. Outline the historical development of software patents.

5. Discuss the consequences of software piracy on software developers and the role of relevant enforcement organizations.

SP7. Privacy and civil liberties [core]

Minimum core coverage time: 2 hours

Topics:
- Ethical and legal basis for privacy protection
- Privacy implications of massive database systems
- Technological strategies for privacy protection
- Freedom of expression in cyberspace
- International and intercultural implications

Learning objectives:

1. Summarize the legal bases for the right to privacy and freedom of expression in one's own nation and how those concepts vary from country to country.

2. Describe current computer-based threats to privacy.

3. Explain how the Internet may change the historical balance in protecting freedom of expression.

4. Explain both the disadvantages and advantages of free expression in cyberspace.

5. Describe trends in privacy protection as exemplified in technology.

5.4 CS 210: Computer Organization (2)(W)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Computer Organiz`

CS 210	**Computer Organization (2)(W)**
new	Digital logic, digital systems, machine-level representation of
required	data, assembly-level machine organization. (Prerequisite: CS 101 or above, Math 110 or above.)

This is derived from, and replaces:

CS 143	**Introduction to Computer Organization (3)(W)**
old	Fundamentals of computer organization and operation. Memory
required	structure, registers, arithmetic and logical functions, instruction format, addressing mode3s, machine and assembly languages, and internal-external data representation. (Prerequisite: Algebra, CS 142.)

CC2001 Knowledge Units: CS 210 will cover the following CC2001 knowledge units: AR1 (digital logic and digital systems, 6h), AR2 (machine-level representation of data, 3h), AR3 (assembly-level machine organization, 9h).

CS 143 is taken by all CS students, typically in their second semester. Part of demystifying what computers do is an examination of how they do it. Binary mathematics is explained and used as a foundation on which all other programming is based.

CS 143 will be scaled back from a three-credit treatment to only two credits. It will be given a new course number, CS 210, but will cover the same core material as before. The promotion of course number from the 100-level to the 200-level reflects our sense of the intermediate level of the material.

CS 210 is a two-credit treatment of material previously presented in a three-credit format in CS 143. It has been given a new course number but will cover the same core material as before. The promotion of course number from the 100-level to the 200-level reflects our sense of the intermediate level of the material.

CS 210 Syllabus (Relevant CC2001 Excerpts)

AR1. Digital logic and digital systems [core]

Minimum core coverage time: 6 hours

Topics:
- Overview and history of computer architecture
- Fundamental building blocks (logic gates, flip-flops, counters, registers, PLA)
- Logic expressions, minimization, sum of product forms
- Register transfer notation
- Physical considerations (gate delays, fan-in, fan-out)

Learning objectives:

1. Describe the progression of computer architecture from vacuum tubes to VLSI.

2. Demonstrate an understanding of the basic building blocks and their role in the historical development of computer architecture.

3. Use mathematical expressions to describe the functions of simple combinational and sequential circuits.

4. Design a simple circuit using the fundamental building blocks.

AR2. Machine level representation of data [core]

Minimum core coverage time: 3 hours

Topics:
- Bits, bytes, and words
- Numeric data representation and number bases
- Fixed- and floating-point systems
- Signed and twos-complement representations
- Representation of nonnumeric data (character codes, graphical data)
- Representation of records and arrays

Learning objectives:

1. Explain the reasons for using different formats to represent numerical data.

2. Explain how negative integers are stored in sign-magnitude and twos-complement representation.

3. Convert numerical data from one format to another.

4. Discuss how fixed-length number representations affect accuracy and precision.

5. Describe the internal representation of nonnumeric data.

6. Describe the internal representation of characters, strings, records, and arrays.

AR3. Assembly level machine organization [core]

Minimum core coverage time: 9 hours

Topics:
- Basic organization of the von Neumann machine
- Control unit; instruction fetch, decode, and execution
- Instruction sets and types (data manipulation, control, I/O)
- Assembly/machine language programming
- Instruction formats
- Addressing modes
- Subroutine call and return mechanisms
- I/O and interrupts

Learning objectives:

1. Explain the organization of the classical von Neumann machine and its major functional units.

2. Explain how an instruction is executed in a classical von Neumann machine.

3. Summarize how instructions are represented at both the machine level and in the context of a symbolic assembler.

4. Explain different instruction formats, such as addresses per instruction and variable length vs. fixed length formats.

5. Write simple assembly language program segments.

6. Demonstrate how fundamental high-level programming constructs are implemented at the machine-language level.

7. Explain how subroutine calls are handled at the assembly level.

8. Explain the basic concepts of interrupts and I/O operations.

5.5 IS 280: Data Communications (2)(F,W,SP)

This course already exists. It is not a new course. It is included here simply to show how it fits into the overall scheme of the CS major.

IS 280	**Data Communications Systems and Networks**
new	**(2)(F,W,SP)**
required	Theory and foundation for network management including data communications hardware and software, network configuration and design, network protocols, and network security and control. (Prerequisite: CS 101.)

This replaces:

CS 460	**Computer Communications and Networking**
old	**(3)(variable)**
elective	Introduction to data communications and computer networking. Communications fundamentals, computer networks, software, architecture, telecommunications, regulation, standards. (Prerequisite: CS 312, 330 or CS 345, 380.)

CC2001 Knowledge Units: IS 280 will cover the following CC2001 knowledge units: NC1 (introduction to net-centric computing, 2h), NC2 (communication and networking, 7h), NC3 (network security, 3h).

The old CS 460 covered both required (core) material and elective (advanced) material. To satisfy the curriculum requirements of accreditation, the core material must be covered for all CS students. Those parts of the course become required.

IS 280 is already taught by the Information Systems department, but will become a CS required course under the proposed plan. Because IS has a perfectly good course that covers core CS material, and IS was willing to guarantee coverage of the knowledge units required by CS, CS elected to not offer its own course in this area, but to direct all its students into the IS course instead.

The advanced (elective) portions of CS 460 disappear for the time being. When we recruit a new faculty member who has expertise in this area, we plan to resurrect this course at the advanced level.

IS 280 Syllabus (Relevant CC2001 Excerpts)

NC1. Introduction to net-centric computing [core]

Minimum core coverage time: 2 hours

Topics:
- Background and history of networking and the Internet
- Network architectures
- The range of specializations within net-centric computing
  - Networks and protocols
  - Networked multimedia systems
  - Distributed computing
  - Mobile and wireless computing

Learning objectives:

1. Discuss the evolution of early networks and the Internet.

2. Demonstrate the ability to use effectively a range of common networked applications including e-mail, telnet, FTP, newsgroups, and web browsers, online web courses, and instant messaging.

3. Explain the hierarchical, layered structure of a typical network architecture.

4. Describe emerging technologies in the net-centric computing area and assess their current capabilities, limitations, and near-term potential.

NC2. Communication and networking [core]

Minimum core coverage time: 7 hours

Topics:
- Network standards and standardization bodies
- The ISO 7-layer reference model in general and its instantiation in TCP/IP
- Circuit switching and packet switching
- Streams and datagrams
- Physical layer networking concepts (theoretical basis, transmission media, standards)
- Data link layer concepts (framing, error control, flow control, protocols)
- Internetworking and routing (routing algorithms, internetworking, congestion control)
- Transport layer services (connection establishment, performance issues)

Learning objectives:

1. Discuss important network standards in their historical context.

2. Describe the responsibilities of the first four layers of the ISO reference model.

3. Discuss the differences between circuit switching and packet switching along with the advantages and disadvantages of each.

4. Explain how a network can detect and correct transmission errors.

5. Illustrate how a packet is routed over the Internet.

6. Install a simple network with two clients and a single server using standard host- configuration software tools such as DHCP.

NC3. Network security [core]

Minimum core coverage time: 3 hours

Topics:
- Fundamentals of cryptography
- Secret-key algorithms
- Public-key algorithms
- Authentication protocols
- Digital signatures
- Examples

Learning objectives:

1. Discuss the fundamental ideas of public-key cryptography.

2. Describe how public-key cryptography works.

3. Distinguish between the use of private- and public-key algorithms.

4. Summarize common authentication protocols.

5. Generate and distribute a PGP key pair and use the PGP package to send an encrypted e-mail message.

6. Summarize the capabilities and limitations of the means of cryptography that are conveniently available to the general public.

5.6 Math 201: Discrete Mathematics I/Lab (2/1)(F,W,SP)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[X] New Course	[ ] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[ ] New Course Descr	[ ] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Discrete Math 1`

Math 201 Discrete Mathematics I (2)(F,W,SP)
new Functions, relations, and sets; basic logic; proof techniques;
required basics of counting. (Prerequisite: Math 110.)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[X] New Course	[ ] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[ ] New Course Descr	[ ] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Discr Math 1 Lab`

Math 201L Discrete Mathematics I Lab (1)(F,W,SP)
new Programming lab work to support Math 201. (Prerequisite:
required CS 101.)

These are derived from, and replace:

CS 235 Foundations of Computer Science I (3)(W)
old Iteration, induction and recursion, lists, trees, sets, relations,
required functions; mathematical analysis of algorithms and data models;
object-oriented implementation of abstract data types.
(Prerequisite: Math 119, CS 142.)

CC2001 Knowledge Units: Math 201 will cover the following CC2001 knowledge units: DS1 (functions, relations, and sets, 6h), DS2 (basic logic, 10h), DS3 (proof techniques, 6/12h), DS4 (basics of counting, 5h).

Math 201 (Discrete 1, theory) and Math 201L (Programming Lab) will be taken by all CS and IS students. CS students would typically take them the second semester of their freshman year. We anticipate seven sections per year with an average enrollment of 30 students per section.

Math 201/L can best be visualized as a team-taught three-credit course with two parts Math and one part CS. The Math component teaches principles and theory and the CS component makes the theory practical through concrete programming experiences. The implementation would be three hours of lecture per week (typically two in Math and one in CS) with an open lab (GCB 101) staffed by student assistants most evenings from 6 PM to midnight. The "lab" section will be taught as 14 hours of lecture, with an expectation of 28 hours of lab work.

The two parts of the course will be graded and credited separately. This allows non-CS students to

take only the math part if they wish, or to audit the CS part. Generally we would expect the CS portion to cover about six programming assignments, requiring on average about six hours each of programming time to properly complete.

Math 201 Syllabus (Relevant CC2001 Excerpts)

DS1. Functions, relations, and sets [core]

Minimum core coverage time: 6 hours

Topics:
- Functions (surjections, injections, inverses, composition)
- Relations (reflexivity, symmetry, transitivity, equivalence relations)
- Sets (Venn diagrams, complements, Cartesian products, power sets)
- Pigeonhole principle
- Cardinality and countability

Learning objectives:

1. Explain with examples the basic terminology of functions, relations, and sets.

2. Perform the operations associated with sets, functions, and relations.

3. Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context.

4. Demonstrate basic counting principles, including uses of diagonalization and the pigeonhole principle.

DS2. Basic logic [core]

Minimum core coverage time: 10 hours

Topics:
- Propositional logic
- Logical connectives
- Truth tables
- Normal forms (conjunctive and disjunctive)
- Validity
- Predicate logic
- Universal and existential quantification
- Modus ponens and modus tollens
- Limitations of predicate logic

Learning objectives:

1. Apply formal methods of symbolic propositional and predicate logic.

2. Describe how formal tools of symbolic logic are used to model algorithms and real- life situations.

3. Use formal logic proofs and logical reasoning to solve problems such as puzzles.

4. Describe the importance and limitations of predicate logic.

DS3. Proof techniques [core]

Minimum core coverage time: 12 hours

Topics:
- Notions of implication, converse, inverse, contrapositive, negation, and contradiction
- The structure of formal proofs
- Direct proofs
- Proof by counterexample
- Proof by contraposition
- Proof by contradiction
- Mathematical induction
- Strong induction
- Recursive mathematical definitions
- Well orderings

Learning objectives:

1. Outline the basic structure of and give examples of each proof technique described in this unit.

2. Discuss which type of proof is best for a given problem.

3. Relate the ideas of mathematical induction to recursion and recursively defined structures.

4. Identify the difference between mathematical and strong induction and give examples of the appropriate use of each.

DS4. Basics of counting [core]

Minimum core coverage time: 5 hours

Topics:
- Counting arguments
  - Sum and product rule
  - Inclusion-exclusion principle
  - Arithmetic and geometric progressions
  - Fibonacci numbers
- The pigeonhole principle
- Permutations and combinations
  - Basic definitions
  - Pascal's identity
  - The binomial theorem
- Solving recurrence relations
  - Common examples
  - The Master theorem

Learning objectives:

1. Compute permutations and combinations of a set, and interpret the meaning in the context of the particular application.

2. State the definition of the Master theorem.

3. Solve a variety of basic recurrence equations.

4. Analyze a problem to create relevant recurrence equations or to identify important counting questions.

5.7 Math 202/L: Discrete Mathematics II/Lab (2/1)(F,W)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[X] New Course	[ ] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[ ] New Course Descr	[ ] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Discrete Math 2`

Math 202 Discrete Mathematics II (2)(F,W)
new Proof techniques (continued), graphs and trees, discrete
required probability. (Prerequisite: Math 201.)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[X] New Course	[ ] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[ ] New Course Descr	[ ] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Discr Math 2 Lab`

Math 202L Discrete Mathematics II Lab (1)(F,W)
new Programming lab work to support Math 202. (Prerequisite:
required Math 201L.)

These are derived from, and replace:

CS 236 Foundations of Computer Science II (3)(F)
old Continuation of CS 235; relations, graphs, automata, grammars,
required propositional and predicate logic. Implementation of
object-oriented algorithms. (Prerequisite: CS 235.)

CC2001 Knowledge Units: Math 202 will cover the following CC2001 knowledge units: DS3 (proof techniques, 6/12h), DS5 (graphs and trees, 4h), DS6 (discrete probability, 6h).

Math 202/L (Discrete 2) will replace CS 236. It will be taken by all CS students, typically in their sophomore year. It is required for a CS minor. We anticipate two sections per year with a total enrollment of 35 to 40 students.

Math 202/L can best be visualized as a team-taught three-credit course with two parts Math and one part CS. The Math component teaches principles and theory and the CS component makes the theory practical through concrete programming experiences. The implementation would be three hours of lecture per week (typically two in Math and one in CS) with an open lab (GCB 101) staffed by student assistants most evenings from 6 PM to midnight. The "lab" section will be taught as 14 hours of lecture, with an expectation of 28 hours of lab work.

The two parts of the course will be graded and credited separately. This allows non-CS students to take only the math part if they wish, or to audit the CS part. Generally we would expect the CS portion to cover about four programming assignments, requiring on average about nine hours each

of programming time to properly complete.

Math 202 Syllabus (Relevant CC2001 Excerpts)

DS3. Proof techniques [core]

Minimum core coverage time: 12 hours

Topics:
- Notions of implication, converse, inverse, contrapositive, negation, and contradiction
- The structure of formal proofs
- Direct proofs
- Proof by counterexample
- Proof by contraposition
- Proof by contradiction
- Mathematical induction
- Strong induction
- Recursive mathematical definitions
- Well orderings

Learning objectives:

1. Outline the basic structure of and give examples of each proof technique described in this unit.

2. Discuss which type of proof is best for a given problem.

3. Relate the ideas of mathematical induction to recursion and recursively defined structures.

4. Identify the difference between mathematical and strong induction and give examples of the appropriate use of each.

DS5. Graphs and trees [core]

Minimum core coverage time: 4 hours

Topics:
- Trees
- Undirected graphs
- Directed graphs
- Spanning trees
- Traversal strategies

Learning objectives:

1. Illustrate by example the basic terminology of graph theory, and some of the properties and special cases of each.

2. Demonstrate different traversal methods for trees and graphs.

3. Model problems in computer science using graphs and trees.

4. Relate graphs and trees to data structures, algorithms, and counting.

DS6. Discrete probability [core]

Minimum core coverage time: 6 hours

Topics:
- Finite probability space, probability measure, events
- Conditional probability, independence, Bayes' theorem
- Integer random variables, expectation

Learning objectives:

1. Calculate probabilities of events and expectations of random variables for elementary problems such as games of chance.

2. Differentiate between dependent and independent events.

3. Apply the binomial theorem to independent events and Bayes theorem to dependent events.

4. Apply the tools of probability to solve problems such as the Monte Carlo method, the average case analysis of algorithms, and hashing.

6 Intermediate Core (Required) Courses

6.1 CS 301: Algorithms and Complexity (3)(F,W)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Algor & Complxty`

CS 301	**Algorithms and Complexity (3)(F,W)**
new	Algorithmic analysis, strategies, fundamental algorithms,
required	distributed algorithms, basic computability. (Prerequisites: CS 202, Math 202/L.)

This is derived from, and replaces:

CS 312	**Algorithm Analysis (3)(F even years)**
old	Analysis of algorithms including searching, sorting, graphs, and
required	trees. (Prerequisites: CS 240, 252.)

CC2001 Knowledge Units: CS 301 will cover the following CC2001 knowledge units: AL1 (basic algorithmic analysis, 4h), AL2 (algorithmic strategies, 6h), AL3 (fundamental computing algorithms, 12h), AL4 (distributed algorithms, 3h), AL5 (basic computability, 6h). AL1 is also covered in CS 201 (Programming II).

CS 301 Syllabus (Relevant CC2001 Excerpts)

AL1. Basic algorithmic analysis [core]

Minimum core coverage time: 4 hours

Topics:
- Asymptotic analysis of upper and average complexity bounds
- Identifying differences among best, average, and worst case behaviors
- Big O, little o, omega, and theta notation
- Standard complexity classes
- Empirical measurements of performance
- Time and space tradeoffs in algorithms
- Using recurrence relations to analyze recursive algorithms

Learning objectives:

1. Explain the use of big O, omega, and theta notation to describe the amount of work done by an algorithm.

2. Use big O, omega, and theta notation to give asymptotic upper, lower, and tight bounds on time and space complexity of algorithms.

3. Determine the time and space complexity of simple algorithms.

4. Deduce recurrence relations that describe the time complexity of recursively defined algorithms.

5. Solve elementary recurrence relations.

AL2. Algorithmic strategies [core]

Minimum core coverage time: 6 hours

Topics:
- Brute-force algorithms
- Greedy algorithms
- Divide-and-conquer
- Backtracking
- Branch-and-bound
- Heuristics
- Pattern matching and string/text algorithms
- Numerical approximation algorithms

Learning objectives:

1. Describe the shortcoming of brute-force algorithms.

2. For each of several kinds of algorithm (brute force, greedy, divide-and-conquer, backtracking, branch-and-bound, and heuristic), identify an example of everyday human behavior that exemplifies

the basic concept.

3. Implement a greedy algorithm to solve an appropriate problem.

4. Implement a divide-and-conquer algorithm to solve an appropriate problem.

5. Use backtracking to solve a problem such as navigating a maze.

6. Describe various heuristic problem-solving methods.

7. Use pattern matching to analyze substrings.

8. Use numerical approximation to solve mathematical problems, such as finding the roots of a polynomial.

AL3. Fundamental computing algorithms [core]

Minimum core coverage time: 12 hours

Topics:
- Simple numerical algorithms
- Sequential and binary search algorithms
- Quadratic sorting algorithms (selection, insertion)
- O(N log N) sorting algorithms (Quicksort, heapsort, mergesort)
- Hash tables, including collision-avoidance strategies
- Binary search trees
- Representations of graphs (adjacency list, adjacency matrix)
- Depth- and breadth-first traversals
- Shortest-path algorithms (Dijkstra's and Floyd's algorithms)
- Transitive closure (Floyd's algorithm)
- Minimum spanning tree (Prim's and Kruskal's algorithms)
- Topological sort

Learning objectives:

1. Implement the most common quadratic and O(N log N) sorting algorithms.

2. Design and implement an appropriate hashing function for an application.

3. Design and implement a collision-resolution algorithm for a hash table.

4. Discuss the computational efficiency of the principal algorithms for sorting, searching, and hashing.

5. Discuss factors other than computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application- specific patterns in the input data.

6. Solve problems using the fundamental graph algorithms, including depth-first and breadth-first search, single-source and all-pairs shortest paths, transitive closure, topological sort, and at least one minimum spanning tree algorithm.

7. Demonstrate the following capabilities: to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in programming context.

AL4. Distributed algorithms [core]

Minimum core coverage time: 3 hours

Topics:
- Consensus and election
- Termination detection
- Fault tolerance
- Stabilization

Learning objectives:

1. Explain the distributed paradigm.

2. Explain one simple distributed algorithm.

3. Determine when to use consensus or election algorithms.

4. Distinguish between logical and physical clocks.

5. Describe the relative ordering of events in a distributed algorithm.

AL5. Basic computability [core]

Minimum core coverage time: 6 hours

Topics:
- Finite-state machines
- Context-free grammars
- Tractable and intractable problems
- Uncomputable functions
- The halting problem
- Implications of uncomputability

Learning objectives:

1. Discuss the concept of finite state machines.

2. Explain context-free grammars.

3. Design a deterministic finite-state machine to accept a specified language.

4. Explain how some problems have no algorithmic solution.

5. Provide examples that illustrate the concept of uncomputability.

6.2 CS 320: Computational Theory (2)(F)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Computtnl Theory`

CS 320	**Introduction to Computational Theory (2)(F)**
new	Finite state automata and regular expressions, context-free
required	grammars and pushdown automata, Turing machines, computability and undecidability. (Prerequisite: CS 201, Math 201.)

This is derived from, and replaces:

CS 252	**Introduction to Computational Theory (3)(W)**
old	Finite state automata and regular expressions, context-free
required	grammars and pushdown automata, Turing machines, computability and undecidability. (Prerequisite: CS 236.)

Theory is required by ABET for accreditation.

CS 320 Syllabus (Relevant CC2001 Excerpts)

AL7. Automata theory [elective]

Topics:

- Deterministic finite automata (DFAs)
- Nondeterministic finite automata (NFAs)
- Equivalence of DFAs and NFAs
- Regular expressions
- The pumping lemma for regular expressions
- Push-down automata (PDAs)
- Relationship of PDAs and context-free grammars
- Properties of context-free grammars
- Turing machines
- Nondeterministic Turing machines
- Sets and languages
- Chomsky hierarchy
- The Church-Turing thesis

Learning objectives:

1. Determine a language's location in the Chomsky hierarchy (regular sets, context-free, context-sensitive, and recursively enumerable languages).

2. Prove that a language is in a specified class and that it is not in the next lower class.

3. Convert among equivalently powerful notations for a language, including among DFAs, NFAs, and regular expressions, and between PDAs and CFGs.

4. Explain at least one algorithm for both top-down and bottom-up parsing.

5. Explain the Church-Turing thesis and its significance.

6.3 CS 333: Software Engineering I (2)(F)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Software Eng 1`

CS 333	**Software Engineering I (2)(F)**
new	Social context of computing, methods and tools of analysis,
required	software design, using APIs, software tools and environments,
	software processes. (Prerequisite: CS 202.)

This is derived from, and partially replaces:

CS 428	**Software System Design and Implementation**
old	**(3)(variable)**
elective	Analysis, design, implementation, and testing of significant
	software systems. (Prerequisite: CS 312, 330 or CS 345, 380.)

The second half is replaced by CS 433.

CC2001 Knowledge Units: CS 333 will cover the following CC2001 knowledge units: SP2 (social context of computing, 3h), SP3 (methods and tools of analysis, 2h), SE1 (software design, 8h), SE2 (using APIs, 5h), SE3 (software tools and environments, 3h), SE4 (software processes, 2h).

CS 333 Syllabus (Relevant CC2001 Excerpts)

SP2. Social context of computing [core]

Minimum core coverage time: 3 hours

Topics:
- Introduction to the social implications of computing
- Social implications of networked communication
- Growth of, control of, and access to the Internet
- Gender-related issues
- International issues

Learning objectives:

1. Interpret the social context of a particular implementation.

2. Identify assumptions and values embedded in a particular design.

3. Evaluate a particular implementation through the use of empirical data.

4. Describe positive and negative ways in which computing alters the modes of interaction between people.

5. Explain why computing/network access is restricted in some countries.

SP3. Methods and tools of analysis [core]

Minimum core coverage time: 2 hours

Topics:
- Making and evaluating ethical arguments
- Identifying and evaluating ethical choices
- Understanding the social context of design
- Identifying assumptions and values

Learning objectives:

1. Analyze an argument to identify premises and conclusion.

2. Illustrate the use of example, analogy, and counter-analogy in ethical argument.

3. Detect use of basic logical fallacies in an argument.

4. Identify stakeholders in an issue and our obligations to them.

5. Articulate the ethical tradeoffs in a technical decision.

SE1. Software design [core]

Minimum core coverage time: 8 hours

Topics:
- Fundamental design concepts and principles
- Design patterns
- Software architecture
- Structured design
- Object-oriented analysis and design
- Component-level design
- Design for reuse

Learning objectives:

1. Discuss the properties of good software design.

2. Compare and contrast object-oriented analysis and design with structured analysis and design.

3. Evaluate the quality of multiple software designs based on key design principles and concepts.

4. Select and apply appropriate design patterns in the construction of a software application.

5. Create and specify the software design for a medium-size software product using a software requirement specification, an accepted program design methodology (e.g., structured or object-oriented), and appropriate design notation.

6. Conduct a software design review using appropriate guidelines.

7. Evaluate a software design at the component level.

8. Evaluate a software design from the perspective of reuse.

SE2. Using APIs [core]

Minimum core coverage time: 5 hours

Topics:
- API programming
- Class browsers and related tools
- Programming by example
- Debugging in the API environment
- Introduction to component-based computing

Learning objectives:

1. Explain the value of application programming interfaces (APIs) in software development.

1. Use class browsers and related tools during the development of applications using APIs.

2. Design, implement, test, and debug programs that use large-scale API packages.

SE3. Software tools and environments [core]

Minimum core coverage time: 3 hours

Topics:
- Programming environments
- Requirements analysis and design modeling tools
- Testing tools
- Configuration management tools
- Tool integration mechanisms

Learning objectives:

1. Select, with justification, an appropriate set of tools to support the development of a range of software products.

2. Analyze and evaluate a set of tools in a given area of software development (e.g., management, modeling, or testing).

3. Demonstrate the capability to use a range of software tools in support of the development of a software product of medium size.

SE4. Software processes [core]

Minimum core coverage time: 2 hours

Topics:
- Software life-cycle and process models
- Process assessment models
- Software process metrics

Learning objectives:

1. Explain the software life cycle and its phases including the deliverables that are produced.

2. Select, with justification the software development models most appropriate for the development and maintenance of a diverse range of software products.

3. Explain the role of process maturity models.

4. Compare the traditional waterfall model to the incremental model, the object-oriented model, and other apropriate models.

5. For each of various software project scenarios, describe the project's place in the software life cycle, identify the particular tasks that should be performed next, and identify metrics appropriate to those tasks.

6.4 IS 351: Advanced Data Files and Data Bases (3)(F,W,SP)

This course already exists. It is not a new course. It is included here simply to show how it fits into the overall scheme of the CS major.

IS 351	**Advanced Data Files and Data Bases (3)(F,W,SP)**
new	Advanced data base management systems. Students expand
required	their working knowledge of data structures, normalization of data, data modeling and data base models within a server environment. (Prerequisite: IS 350 or CS 202.)

This replaces:

CS 452	**Database Modeling Concepts (3)(variable)**
old	Database models: relational, network, hierarchical, deductive,
elective	object-oriented. Integrity constraints, query languages, database design. (Prerequisite: CS 312, 330 or CS 345, 380.)

CC2001 Knowledge Units: IS 351 will cover the following CC2001 knowledge units: IM1 (information models and systems, 3h), IM2 (database systems, 3h), IM3 (data modeling, 4h), IM4 (relational databases, non-core), IM5 (database query languages, non-core), IM6 (relational database design, non-core).

IS 351 is a new course that will be taught by the Information Systems department, but will become a CS required course under the proposed plan. Because IS is planning a course that covers core CS material, and IS has promised coverage of the knowledge units required by CS, CS elected to not offer its own course in this area, but to direct all its students into the IS course instead.

IS 351 Syllabus (Relevant CC2001 Excerpts)

IM1. Information models and systems [core]

Minimum core coverage time: 3 hours

Topics:
- History and motivation for information systems
- Information storage and retrieval (IS&R)
- Information management applications
- Information capture and representation
- Analysis and indexing
- Search, retrieval, linking, navigation
- Information privacy, integrity, security, and preservation
- Scalability, efficiency, and effectiveness

Learning objectives:

1. Compare and contrast information with data and knowledge.

2. Summarize the evolution of information systems from early visions up through modern offerings, distinguishing their respective capabilities and future potential.

3. Critique/defend a small- to medium-size information application with regard to its satisfying real user information needs.

4. Describe several technical solutions to the problems related to information privacy, integrity, security, and preservation.

5. Explain measures of efficiency (throughput, response time) and effectiveness (recall, precision).

6. Describe approaches to ensure that information systems can scale from the individual to the global.

IM2. Database systems [core]

Minimum core coverage time: 3 hours

Topics:
- History and motivation for database systems
- Components of database systems
- DBMS functions
- Database architecture and data independence
- Use of a database query language

Learning objectives:

1. Explain the characteristics that distinguish the database approach from the traditional approach of programming with data files.

2. Cite the basic goals, functions, models, components, applications, and social impact of database systems.

3. Describe the components of a database system and give examples of their use.

4. Identify major DBMS functions and describe their role in a database system.

5. Explain the concept of data independence and its importance in a database system.

6. Use a query language to elicit information from a database.

IM3. Data modeling [core]

Minimum core coverage time: 4 hours

Topics:
- Data modeling
- Conceptual models (including entity-relationship and UML)
- Object-oriented model
- Relational data model

Learning objectives:

1. Categorize data models based on the types of concepts that they provide to describe the database structure-that is, conceptual data model, physical data model, and representational data model.

2. Describe the modeling concepts and notation of the entity-relationship model and UML, including their use in data modeling.

3. Describe the main concepts of the OO model such as object identity, type constructors, encapsulation, inheritance, polymorphism, and versioning.

4. Define the fundamental terminology used in the relational data model .

5. Describe the basic principles of the relational data model.

6. Illustrate the modeling concepts and notation of the relational data model.

IM4. Relational databases [elective]

Topics:
- Mapping conceptual schema to a relational schema
- Entity and referential integrity
- Relational algebra and relational calculus

Learning objectives:

1. Prepare a relational schema from a conceptual model developed using the entity- relationship model

2. Explain and demonstrate the concepts of entity integrity constraint and referential integrity

constraint (including definition of the concept of a foreign key).

3. Demonstrate use of the relational algebra operations from mathematical set theory (union, intersection, difference, and cartesian product) and the relational algebra operations developed specifically for relational databases (select, product, join, and division).

4. Demonstrate queries in the relational algebra.

5. Demonstrate queries in the tuple relational calculus.

IM5. Database query languages [elective]

Topics:
- Overview of database languages
- SQL (data definition, query formulation, update sublanguage, constraints, integrity)
- Query optimization
- QBE and 4th-generation environments
- Embedding non-procedural queries in a procedural language
- Introduction to Object Query Language

Learning objectives:

1. Create a relational database schema in SQL that incorporates key, entity integrity, and referential integrity constraints.

2. Demonstrate data definition in SQL and retrieving information from a database using the SQL SELECT statement.

3. Evaluate a set of query processing strategies and select the optimal strategy.

4. Create a non-procedural query by filling in templates of relations to construct an example of the desired query result.

5. Embed object-oriented queries into a stand-alone language such as C++ or Java (e.g., SELECT Col.Method() FROM Object).

IM6. Relational database design [elective]

Topics:
- Database design
- Functional dependency
- Normal forms (1NF, 2NF, 3NF, BCNF)
- Multivalued dependency (4NF)
- Join dependency (PJNF, 5NF)
- Representation theory

Learning objectives:

1. Determine the functional dependency between two or more attributes that are a subset of a

relation.

2. Describe what is meant by 1NF, 2NF, 3NF, and BCNF.

3. Identify whether a relation is in 1NF, 2NF, 3NF, or BCNF.

4. Normalize a 1NF relation into a set of 3NF (or BCNF) relations and denormalize a relational schema.

5. Explain the impact of normalization on the efficiency of database operations, especially query optimization.

6. Describe what is a multivalued dependency and what type of constraints it specifies.

7. Explain why 4NF is useful in schema design.

7 Intermediate Non-Core (Elective) Courses

7.1 CS 390R: Special Topics in Computer Science (1-3)(var)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Spec Topics CS`

CS 390R	**Special Topics in Computer Science (1-3)(variable)**
new	Selected intermediate topics in Computer Science. (Prerequisite: instructor's consent.)

This replaces:

CS 390R	**Special Topics in Computer Science (1-3)(variable)**
old	Selected topics in Computer Science.

The only change is the insertion of the word "intermediate" into the course description.

For accreditation purposes, the CS curriculum must clearly distinguish between courses that are intermediate and those that are advanced. We do this by designating 300-level courses as intermediate, and 400-level courses as advanced.

A matching new course, CS 490R, Advanced Topics in CS is proposed to allow CS to offer experimental courses in the advanced category.

7.2 CS 399R: Cooperative Education in Computer Science (1-3)

Abbreviated Title (16 char): `Coop Edu in CS`

CS 399R **Cooperative Education in Computer Science (1-3)**
Students may receive credit for on-the-job experience in Computer Science. Prior approval must be obtained and a program worked out between the faculty supervisor and the employer to determine what the student is expected to accomplish. Credit is earned from the learning which takes place, not from the work performed.

No changes are proposed for this course. It remains as it was.

8 Advanced Core (Required) Courses

8.1 CS 410: Computer Architecture (2)(F)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Compu Architec`

CS 410	**Computer Architecture (2)(F)**
new	Memory system organization and architecture, interfacing and
required	communication, functional organization, multiprocessing and alternative architectures. (Prerequisite: CS 210, CS 301.)

This is derived from, and replaces:

CS 380	**Computer Architecture (3)(F odd years)**
old	Instruction sets, hardwired and microprogrammed control,
required	interrupts, memory hierarchy, virtual memory, input/output interfacing, and alternative architecture. (Prerequisites: CS 143, 240.)

CC2001 Knowledge Units: CS 410 will cover the following CC2001 knowledge units: AR4 (memory system organization and architecture, 5h), AR5 (interfacing and communication, 3h), AR6 (functional organization, 7h), AR7 (multiprocessing and alternative architectures, 3h).

The main change is a reduction in credit hours from three to two, with a commensurate reduction in the material covered. At the same time we are updating the course content.

The promotion of course number from the 300-level to the 400-level reflects our sense of the advanced level of the material.

CS 410 Syllabus (Relevant CC2001 Excerpts)

AR4. Memory system organization and architecture [core]

Minimum core coverage time: 5 hours

Topics:
- Storage systems and their technology
- Coding, data compression, and data integrity
- Memory hierarchy
- Main memory organization and operations
- Latency, cycle time, bandwidth, and interleaving
- Cache memories (address mapping, block size, replacement and store policy)
- Virtual memory (page table, TLB)
- Fault handling and reliability

Learning objectives:

1. Identify the main types of memory technology.
2. Explain the effect of memory latency on running time.
3. Explain the use of memory hierarchy to reduce the effective memory latency.
4. Describe the principles of memory management.
5. Describe the role of cache and virtual memory.
6. Explain the workings of a system with virtual memory management.

AR5. Interfacing and communication [core]

Minimum core coverage time: 3 hours

Topics:
- I/O fundamentals: handshaking, buffering, programmed I/O, interrupt-driven I/O
- Interrupt structures: vectored and prioritized, interrupt acknowledgment
- External storage, physical organization, and drives
- Buses: bus protocols, arbitration, direct-memory access (DMA)
- Introduction to networks
- Multimedia support
- RAID architectures

Learning objectives:

1. Explain how interrupts are used to implement I/O control and data transfers.
2. Identify various types of buses in a computer system.

3. Describe data access from a magnetic disk drive.

4. Compare the common network configurations.

5. Identify interfaces needed for multimedia support.

6. Describe the advantages and limitations of RAID architectures.

AR6. Functional organization [core]

Minimum core coverage time: 7 hours

Topics:
- Implementation of simple datapaths
- Control unit: hardwired realization vs. microprogrammed realization
- Instruction pipelining
- Introduction to instruction-level parallelism (ILP)

Learning objectives:

1. Compare alternative implementation of datapaths.

2. Discuss the concept of control points and the generation of control signals using hardwired or microprogrammed implementations.

3. Explain basic instruction level parallelism using pipelining and the major hazards that may occur.

AR7. Multiprocessing and alternative architectures [core]

Minimum core coverage time: 3 hours

Topics:
- Introduction to SIMD, MIMD, VLIW, EPIC
- Systolic architecture
- Interconnection networks (hypercube, shuffle-exchange, mesh, crossbar)
- Shared memory systems
- Cache coherence
- Memory models and memory consistency

Learning objectives:

1. Discuss the concept of parallel processing beyond the classical von Neumann model.

2. Describe alternative architectures such as SIMD, MIMD, and VLIW.

3. Explain the concept of interconnection networks and characterize different approaches.

4. Discuss the special concerns that multiprocessing systems present with respect to memory management and describe how these are addressed.

8.2 CS 415: Operating Systems Design (2)(F)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[ ] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Oper Sys Design`

CS 415	**Operating Systems Design (2)(F)**
new	Operating system principles, concurrency, scheduling, dispatch,
required	memory management. (Prerequisite: CS 210, CS 301.)

This is derived from, and replaces:

CS 345	**Operating Systems Design (3)(F odd years)**
old	Principles and concepts of operating systems design and the
required	implementation of an operating system. (Prerequisite: CS 380 or concurrent registration.)

CC2001 Knowledge Units: CS 415 will cover the following CC2001 knowledge units: OS1 (overview of operating systems, 2h), OS2 (operating system principles, 2h), OS3 (concurrency, 6h), OS4 (scheduling and dispatch, 3h), OS5 (memory management, 5h).

The main change is a reduction in credit hours from three to two, with a commensurate reduction in the material covered.

CS 415 is a two-credit treatment of material previously presented in a three-credit format in CS 345. It has been given a new course number but will cover the same core material as before. The promotion of course number from the 300-level to the 400-level reflects our sense of the advanced level of the material.

CS 415 Syllabus (Relevant CC2001 Excerpts)

OS1. Overview of operating systems [core]

Minimum core coverage time: 2 hours

Topics:
- Role and purpose of the operating system
- History of operating system development
- Functionality of a typical operating system
- Mechanisms to support client-server models, hand-held devices
- Design issues (efficiency, robustness, flexibility, portability, security, compatibility)
- Influences of security, networking, multimedia, windows

Learning objectives:

1. Explain the objectives and functions of modern operating systems.

2. Describe how operating systems have evolved over time from primitive batch systems to sophisticated multiuser systems.

3. Analyze the tradeoffs inherent in operating system design.

4. Describe the functions of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve.

5. Discuss networked, client-server, distributed operating systems and how they differ from single user operating systems.

6. Identify potential threats to operating systems and the security features design to guard against them.

7. Describe how issues such as open source software and the increased use of the Internet are influencing operating system design.

OS2. Operating system principles [core]

Minimum core coverage time: 2 hours

Topics:
- Structuring methods (monolithic, layered, modular, micro-kernel models)
- Abstractions, processes, and resources
- Concepts of application program interfaces (APIs)
- Application needs and the evolution of hardware/software techniques
- Device organization
- Interrupts: methods and implementations
- Concept of user/system state and protection, transition to kernel mode

Learning objectives:

1. Explain the concept of a logical layer.

2. Explain the benefits of building abstract layers in hierarchical fashion.

3. Defend the need for APIs and middleware.

4. Describe how computing resources are used by application software and managed by system software.

5. Contrast kernel and user mode in an operating system.

6. Discuss the advantages and disadvantages of using interrupt processing.

7. Compare and contrast the various ways of structuring an operating system such as object-oriented, modular, micro-kernel, and layered.

8. Explain the use of a device list and driver I/O queue.

OS3. Concurrency [core]

Minimum core coverage time: 6 hours

Topics:
- States and state diagrams
- Structures (ready list, process control blocks, and so forth)
- Dispatching and context switching
- The role of interrupts
- Concurrent execution: advantages and disadvantages
- The "mutual exclusion" problem and some solutions
- Deadlock: causes, conditions, prevention
- Models and mechanisms (semaphores, monitors, condition variables, rendezvous)
- Producer-consumer problems and synchronization
- Multiprocessor issues (spin-locks, reentrancy)

Learning objectives:

1. Describe the need for concurrency within the framework of an operating system.

2. Demonstrate the potential run-time problems arising from the concurrent operation of many separate tasks.

3. Summarize the range of mechanisms that can be employed at the operating system level to realize concurrent systems and describe the benefits of each.

4. Explain the different states that a task may pass through and the data structures needed to support the management of many tasks.

5. Summarize the various approaches to solving the problem of mutual exclusion in an operating system.

6. Describe reasons for using interrupts, dispatching, and context switching to support concurrency in an operating system.

7. Create state and transition diagrams for simple problem domains.

8. Discuss the utility of data structures, such as stacks and queues, in managing concurrency.

9. Explain conditions that lead to deadlock.

OS4. Scheduling and dispatch [core]

Minimum core coverage time: 3 hours

Topics:
- Preemptive and nonpreemptive scheduling
- Schedulers and policies
- Processes and threads
- Deadlines and real-time issues

Learning objectives:

1. Compare and contrast the common algorithms used for both preemptive and non- preemptive scheduling of tasks in operating systems, such as priority, performance comparison, and fair-share schemes.

2. Describe relationships between scheduling algorithms and application domains.

3. Discuss the types of processor scheduling such as short-term, medium-term, long- term, and I/O.

4. Describe the difference between processes and threads.

5. Compare and contrast static and dynamic approaches to real-time scheduling.

6. Discuss the need for preemption and deadline scheduling.

7. Identify ways that the logic embodied in scheduling algorithms are applicable to other domains, such as disk I/O, network scheduling, project scheduling, and other problems unrelated to computing.

OS5. Memory management [core]

Minimum core coverage time: 5 hours

Topics:
- Review of physical memory and memory management hardware
- Overlays, swapping, and partitions
- Paging and segmentation
- Placement and replacement policies
- Working sets and thrashing
- Caching

Learning objectives:

1. Explain memory hierarchy and cost-performance tradeoffs.

2. Explain the concept of virtual memory and how it is realized in hardware and software.

3. Summarize the principles of virtual memory as applied to caching, paging, and segmentation.

4. Evaluate the tradeoffs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed.

5. Defend the different ways of allocating memory to tasks, citing the relative merits of each.

6. Describe the reason for and use of cache memory.

7. Compare and contrast paging and segmentation techniques.

8. Discuss the concept of thrashing, both in terms of the reasons it occurs and the techniques used to recognize and manage the problem.

9. Analyze the various memory portioning techniques including overlays, swapping, and placement and replacement policies.

8.3 CS 420: Programming Languages (2)(W)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Prog Languages`

CS 420	**Programming Languages (2)(W)**
new	Virtual machines, language translation, declarations and types,
required	abstraction mechanisms, object-oriented programming.
	(Prerequisite: CS 202 and CS 320.)

This is derived from, and replaces:

CS 330	**Concepts of Programming Languages (3)(F even years)**
old	Principles and concepts characterizing high-level computer
required	programming languages, process and data abstraction,
	encapsulation, inheritance, functional programming, logic
	programming, scanners, parsers. (Prerequisites: CS 240, 252.)

CC2001 Knowledge Units: CS 420 will cover the following CC2001 knowledge units: PL1 (overview of programming languages, 2h), PL2 (virtual machines, 1h), PL3 (introduction to language translation, 2h), PL4 (declarations and types, 3h), PL5 (abstraction mechanisms, 3h), PL6 (object-oriented programming, 10h).

CS 330 will be scaled back from a three-credit treatment to only two credits. It will be given a new course number, CS 420, but will cover the same core material as before.

CS 420 Syllabus (Relevant CC2001 Excerpts)

PL1. Overview of programming languages [core]

Minimum core coverage time: 2 hours

Topics:
- History of programming languages
- Brief survey of programming paradigms
  - Procedural languages
  - Object-oriented languages
  - Functional languages
  - Declarative, non-algorithmic languages
  - Scripting languages
- The effects of scale on programming methodology

Learning objectives:

1. Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today.

2. Identify at least one distinguishing characteristic for each of the programming paradigms covered in this unit.

3. Evaluate the tradeoffs between the different paradigms, considering such issues as space efficiency, time efficiency (of both the computer and the programmer), safety, and power of expression.

4. Distinguish between programming-in-the-small and programming-in-the-large.

PL2. Virtual machines [core]

Minimum core coverage time: 1 hour

Topics:
- The concept of a virtual machine
- Hierarchy of virtual machines
- Intermediate languages
- Security issues arising from running code on an alien machine

Learning objectives:

1. Describe the importance and power of abstraction in the context of virtual machines.

2. Explain the benefits of intermediate languages in the compilation process.

3. Evaluate the tradeoffs in performance vs. portability.

4. Explain how executable programs can breach computer system security by accessing disk files and memory.

PL3. Introduction to language translation [core]

Minimum core coverage time: 2 hours

Topics:
- Comparison of interpreters and compilers
- Language translation phases (lexical analysis, parsing, code generation, optimization)
- Machine-dependent and machine-independent aspects of translation

Learning objectives:

1. Compare and contrast compiled and interpreted execution models, outlining the relative merits of each..

2. Describe the phases of program translation from source code to executable code and the files produced by these phases.

3. Explain the differences between machine-dependent and machine-independent translation and where these differences are evident in the translation process.

PL4. Declarations and types [core]

Minimum core coverage time: 3 hours

Topics:
- The conception of types as a set of values with together with a set of operations
- Declaration models (binding, visibility, scope, and lifetime)
- Overview of type-checking
- Garbage collection

Learning objectives:

1. Explain the value of declaration models, especially with respect to programming-in- the-large.

2. Identify and describe the properties of a variable such as its associated address, value, scope, persistence, and size.

3. Discuss type incompatibility.

4. Demonstrate different forms of binding, visibility, scoping, and lifetime management.

5. Defend the importance of types and type-checking in providing abstraction and safety.

6. Evaluate tradeoffs in lifetime management (reference counting vs. garbage collection).

PL5. Abstraction mechanisms [core]

Minimum core coverage time: 3 hours

Topics:
- Procedures, functions, and iterators as abstraction mechanisms

- Parameterization mechanisms (reference vs. value)
- Activation records and storage management
- Type parameters and parameterized types
- Modules in programming languages

Learning objectives:

1. Explain how abstraction mechanisms support the creation of reusable software components.

2. Demonstrate the difference between call-by-value and call-by-reference parameter passing.

3. Defend the importance of abstractions, especially with respect to programming-in- the-large.

4. Describe how the computer system uses activation records to manage program modules and their data.

PL6. Object-oriented programming [core]

Minimum core coverage time: 10 hours

Topics:
- Object-oriented design
- Encapsulation and information-hiding
- Separation of behavior and implementation
- Classes and subclasses
- Inheritance (overriding, dynamic dispatch)
- Polymorphism (subtype polymorphism vs. inheritance)
- Class hierarchies
- Collection classes and iteration protocols
- Internal representations of objects and method tables

Learning objectives:

1. Justify the philosophy of object-oriented design and the concepts of encapsulation, abstraction, inheritance, and polymorphism.

2. Design, implement, test, and debug simple programs in an object-oriented programming language.

3. Describe how the class mechanism supports encapsulation and information hiding.

4. Design, implement, and test the implementation of "is-a" relationships among objects using a class hierarchy and inheritance.

5. Compare and contrast the notions of overloading and overriding methods in an object-oriented language.

6. Explain the relationship between the static structure of the class and the dynamic structure of the instances of the class.

7. Describe how iterators access the elements of a container.

8.4 CS 433: Software Engineering II (2)(W)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Software Eng 2`

CS 433	**Software Engineering II (2)(W)**
new	Software requirements and specifications, validation, evolution,
required	project management, foundations of human-computer interaction, building a simple graphical user interface, fundamental techniques in graphics, graphic systems. (Prerequisite: CS 333.)

This is derived from, and partially replaces:

CS 428	**Software System Design and Implementation**
old	**(3)(variable)**
elective	Analysis, design, implementation, and testing of significant software systems. (Prerequisite: CS 312, 330 or CS 345, 380.)

The first half was replaced by CS 333.

CC2001 Knowledge Units: CS 433 will cover the following CC2001 knowledge units: SE5 (software requirements and specifications, 4h), SE6 (software validation, 3h), SE7 (software evolution, 3h), SE8 (software project management, 3h), HC1 (foundations of human-computer interaction, 6h), HC2 (building a simple graphical user interface, 2h), GV1 (fundamental techniques in graphics, 2h), GV2 (graphic systems, 1h).

CS 433 Syllabus (Relevant CC2001 Excerpts)

SE5. Software requirements and specifications [core]

Minimum core coverage time: 4 hours

Topics:
- Requirements elicitation
- Requirements analysis modeling techniques
- Functional and nonfunctional requirements
- Prototyping
- Basic concepts of formal specification techniques

Learning objectives:

1. Apply key elements and common methods for elicitation and analysis to produce a set of software requirements for a medium-sized software system.

2. Discuss the challenges of maintaining legacy software.

3. Use a common, non-formal method to model and specify (in the form of a requirements specification document) the requirements for a medium-size software system.

4. Conduct a review of a software requirements document using best practices to determine the quality of the document.

5. Translate into natural language a software requirements specification written in a commonly used formal specification language.

SE6. Software validation [core]

Minimum core coverage time: 3 hours

Topics:
- Validation planning
- Testing fundamentals, including test plan creation and test case generation
- Black-box and white-box testing techniques
- Unit, integration, validation, and system testing
- Object-oriented testing
- Inspections

Learning objectives:

1. Distinguish between program validation and verification.

2. Describe the role that tools can play in the validation of software.

3. Distinguish between the different types and levels of testing (unit, integration, systems, and acceptance) for medium-size software products.

4. Create, evaluate, and implement a test plan for a medium-size code segment.

5. Undertake, as part of a team activity, an inspection of a medium-size code segment.

6. Discuss the issues involving the testing of object-oriented software.

SE7. Software evolution [core]

Minimum core coverage time: 3 hours

Topics:
- Software maintenance
- Characteristics of maintainable software
- Reengineering
- Legacy systems
- Software reuse

Learning objectives:

1. Identify the principal issues associated with software evolution and explain their impact on the software life cycle.

2. Discuss the challenges of maintaining legacy systems and the need for reverse engineering.

3. Outline the process of regression testing and its role in release management.

4. Estimate the impact of a change request to an existing product of medium size.

5. Develop a plan for re-engineering a medium-sized product in response to a change request.

6. Discuss the advantages and disadvantages of software reuse.

7. Exploit opportunities for software reuse in a given context.

SE8. Software project management [core]

Minimum core coverage time: 3 hours

Topics:
- Team management
  - Team processes
  - Team organization and decision-making
  - Roles and responsibilities in a software team
  - Role identification and assignment
  - Project tracking
  - Team problem resolution
- Project scheduling
- Software measurement and estimation techniques
- Risk analysis
- Software quality assurance

- Software configuration management
- Project management tools

Learning objectives:

1. Demonstrate through involvement in a team project the central elements of team building and team management.

2. Prepare a project plan for a software project that includes estimates of size and effort, a schedule, resource allocation, configuration control, change management, and project risk identification and management.

3. Compare and contrast the different methods and techniques used to assure the quality of a software product.

HC1. Foundations of human-computer interaction [core]

Minimum core coverage time: 6 hours

Topics:
- Motivation: Why care about people?
- Contexts for HCI (tools, web hypermedia, communication)
- Human-centered development and evaluation
- Human performance models: perception, movement, and cognition
- Human performance models: culture, communication, and organizations
- Accommodating human diversity
- Principles of good design and good designers; engineering tradeoffs
- Introduction to usability testing

Learning objectives:

1. Discuss the reasons for human-centered software development.

2. Summarize the basic science of psychological and social interaction.

3. Differentiate between the role of hypotheses and experimental results vs. correlations.

4. Develop a conceptual vocabulary for analyzing human interaction with software: affordance, conceptual model, feedback, and so forth.

5. Distinguish between the different interpretations that a given icon, symbol, word, or color can have in (a) two different human cultures and (b) in a culture and one of its subcultures.

6. In what ways might the design of a computer system or application succeed or fail in terms of respecting human diversity.

7. Create and conduct a simple usability test for an existing software application.

HC2. Building a simple graphical user interface [core]

Minimum core coverage time: 2 hours

Topics:
- Principles of graphical user interfaces (GUIs)
- GUI toolkits

Learning objectives:

1. Identify several fundamental principles for effective GUI design.

2. Use a GUI toolkit to create a simple application that supports a graphical user interface.

3. Illustrate the effect of fundamental design principles on the structure of a graphical user interface.

4. Conduct a simple usability test for each instance and compare the results.

GV1. Fundamental techniques in graphics [core]

Minimum core coverage time: 2 hours

Topics:
- Hierarchy of graphics software
- Using a graphics API
- Simple color models (RGB, HSB, CMYK)
- Homogeneous coordinates
- Affine transformations (scaling, rotation, translation)
- Viewing transformation
- Clipping

Learning objectives:

1. Distinguish the capabilities of different levels of graphics software and describe the appropriateness of each.

2. Create images using a standard graphics API.

3. Use the facilities provided by a standard API to express basic transformations such as scaling, rotation, and translation.

4. Implement simple procedures that perform transformation and clipping operations on a simple 2-dimensional image.

5. Discuss the 3-dimensional coordinate system and the changes required to extend 2D transformation operations to handle transformations in 3D

GV2. Graphic systems [core]

Minimum core coverage time: 1 hour

Topics:

- Raster and vector graphics systems
- Video display devices
- Physical and logical input devices
- Issues facing the developer of graphical systems

Learning objectives:

1. Describe the appropriateness of graphics architectures for given applications.

2. Explain the function of various input devices.

3. Compare and contrast the techniques of raster graphics and vector graphics.

4. Use current hardware and software for creating and displaying graphics.

5. Discuss the expanded capabilities of emerging hardware and software for creating and displaying graphics.

8.5 CS 440: Intelligent Systems (2)(W)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Intellig Systems`

CS 440	**Intelligent Systems (2)(W)**
new	Fundamental issues in intelligent systems, search and constraint
required	satisfaction, knowledge representation and reasoning.
	(Prerequisite: CS 301.)

This is derived from, and replaces:

CS 470	**Introduction to Artificial Intelligence (3)(variable)**
old	Introduction to core areas of artificial intelligence; intelligent
elective	agents, problem solving and search, knowledge-based systems
	and inference, planning, uncertainty, learning, and perception.
	(Prerequisite: CS 312, 330 or CS 345, 380.)

CC2001 Knowledge Units: CS 440 will cover the following CC2001 knowledge units: IS1 (fundamental issues in intelligent systems, 1h), IS2 (search and constraint satisfaction, 5h), IS3 (knowledge representation and reasoning, 4h).

The main change is a reduction in credit hours from three to two, with a commensurate reduction in the material covered. At the same time we are updating the course content.

CS 440 Syllabus (Relevant CC2001 Excerpts)

IS1. Fundamental issues in intelligent systems [core]

Minimum core coverage time: 1 hour

Topics:
- History of artificial intelligence
- Philosophical questions
  - The Turing test
  - Searle's "Chinese Room" thought experiment
  - Ethical issues in AI
- Fundamental definitions
  - Optimal vs. human-like reasoning
  - Optimal vs. human-like behavior
- Philosophical questions
- Modeling the world
- The role of heuristics

Learning objectives:

1. Describe the Turing test and the "Chinese Room" thought experiment.

2. Differentiate the concepts of optimal reasoning and human-like reasoning.

3. Differentiate the concepts of optimal behavior and human-like behavior.

4. List examples of intelligent systems that depend on models of the world.

5. Describe the role of heuristics and the need for tradeoffs between optimality and efficiency.

IS2. Search and constraint satisfaction [core]

Minimum core coverage time: 5 hours

Topics:
- Problem spaces
- Brute-force search (breadth-first, depth-first, depth-first with iterative deepening)
- Best-first search (generic best-first, Dijkstra's algorithm, A*, admissibility of A*)
- Two-player games (minimax search, alpha-beta pruning)
- Constraint satisfaction (backtracking and local search methods)

Learning objectives:

1. Formulate an efficient problem space for a problem expressed in English by expressing that problem space in terms of states, operators, an initial state, and a description of a goal state.

2. Describe the problem of combinatorial explosion and its consequences.

3. Select an appropriate brute-force search algorithm for a problem, implement it, and characterize its time and space complexities.

4. Select an appropriate heuristic search algorithm for a problem and implement it by designing the necessary heuristic evaluation function.

5. Describe under what conditions heuristic algorithms guarantee optimal solution.

6. Implement minimax search with alpha-beta pruning for some two-player game.

7. Formulate a problem specified in English as a constraint-satisfaction problem and implement it using a chronological backtracking algorithm.

IS3. Knowledge representation and reasoning [core]

Minimum core coverage time: 4 hours

Topics:
- Review of propositional and predicate logic
- Resolution and theorem proving
- Nonmonotonic inference
- Probabilistic reasoning
- Bayes theorem

Learning objectives:

1. Explain the operation of the resolution technique for theorem proving.

2. Explain the distinction between monotonic and nonmonotonic inference.

3. Discuss the advantages and shortcomings of probabilistic reasoning.

4. Apply Bayes theorem to determine conditional probabilities.

9 Advanced Non-Core (Elective) Courses

9.1 CS 421: Algorithmic Languages and Compilers (2)(var)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[ ] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[ ] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Alg L/Compilers`

CS 421	**Algorithmic Languages and Compilers (2)(variable)**
new	Formal description of algorithmic languages and techniques used
elective	in their compilation: semantics, ambiguities, procedures, replication, iteration, recursion. Design and implementation of a simple compiler. (Prerequisite: CS 420.)

This is derived from, and replaces:

CS 431	**Algorithmic Languages and Compilers (3)(variable)**
old	Formal description of algorithmic languages and techniques used
elective	in their compilation: semantics, ambiguities, procedures, replication, iteration, recursion. Design and implementation of a simple compiler. (Prerequisite: CS 312, 330 or CS 345, 380.)

The main change is a reduction in credit hours from three to two, with a commensurate reduction in the material covered.

9.2 CS 441: Automatic Speech Recognition (2)(var)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[X] New Course	[ ] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[ ] New Course Descr	[ ] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Aut Speech Recog`

CS 441	**Automatic Speech Recognition (2)(variable)**
new	Introduction to automatic speech recognition by computers,
elective	including digital sampling, Fourier transformation, phonemic classification by neural network, and Viterbi search. (Prerequisite: CS 440.)

This is a new elective. It is an area of research specialization for Brother Colton. He has taught graduate-level courses in automatic speech recognition several times before at Oregon Graduate Institute / Oregon Health and Science University.

It is recognized that undergraduate students, even at the senior level, will not be prepared to complete the course at a graduate level. It is intended that the first time through the course, adjustment will be made to the pace, quantity, and depth of the material to match the abilities of the students in the course.

Please see the next page for a syllabus.

CS 441 Syllabus:

Topics:

- Phonetic Representation of Speech.
  - Phonetic Transcription.
- Speech Production: Vocal Resonances.
- Formants.
- Spectrogram Reading.
- Digital Sampling of Sound.
- Fourier Data Transformation.
- Neural Network Training.
- String Matching by Viterbi Search.

Learning objectives:

1. Given a written sentence, provide a phonetic rendering according to one's own speech habits.

2. Transcribe a recorded utterance into IPA (the International Phonetic Alphabet).

3. Identify and explain the major resonances in the human speech production organs.

4. Explain formants.

5. Given a spectrogram (time-frequency chart) of a human speech utterance, decode it to identify first the phonemes of the speech and second the full text of the utterance.

6. Implement a computer program to convert digitized human speech into spectrogram format.

7. Train a neural network to recognize phonemes.

8. Given hypothetical neural network scores for the frames of a speech utterance, calculate by hand the Viterbi score of two alternate phrases and identify the more correct phrase.

9.3 CS 442: Image Processing and Computer Vision (2)(var)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[X] New Course	[ ] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[ ] New Course Descr	[ ] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Img Proc Cmp Vsn`

CS 442	**Image Processing and Computer Vision (2)(variable)**
new	Techniques for the processing and automatic recognition of
elective	objects and groups of objects in digital images. (Prerequisite: CS 440.)

This is a new elective. It is an area of research specialization for Brother Fife.

Please see the next page for a syllabus.

CS 456 Syllabus:

This course will introduce to the student to the basic principles of image processing. These principles will be used to investigate some simple problems in computer vision (machine vision, robot vision, face and gesture recognition, etc.).

Possible Texts:

- Computer Vision, 1 ed., L. Shapiro and G. Stockman, Prentice Hall.
- Introductory Techniques for 3-D Computer Vision, E. Trucco and A. Verri, Prentice Hall

Topics:

- Imaging and Image Representation (1 week)
- Binary Image Analysis (7 weeks)
  - Histogram
  - thresholding
  - edge detection
  - image segmentation
  - object grouping
- Image Characteristics (2-3 weeks)
  - Shading
  - texture
  - motion
  - optical flow
- Pattern Recognition Concepts. (2-3 weeks)
  - labeling object groups
  - matching
  - rotation, translation and scaling.
- Advanced topics - as time permits
  - Color
  - stereo vision
  - 3-D models
  - pose computation
  - virtual reality
- Case study - to follow throughout the course.

9.4 CS 451: Advanced Database Topics (2)(var)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[X] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Adv D B Topics`

CS 451	**Advanced Database Topics (2)(variable)**
new	Specialized topics in database such as data mining, data
elective	warehousing, intelligent database systems, object-oriented database systems, and emerging topics in database research. (Prerequisite: CS 301 and IS 351.)

This replaces:

CS 453	**Advanced Data Structures (3)(variable)**
old	File organization and management, external data structures,
elective	database implementation, including concurrency control, transaction processing, and distributed databases. (Prerequisite: CS 312, 330 or CS 345, 380.)

The main change is a reduction in credit hours from three to two, with a commensurate reduction in the material covered. At the same time we are updating the course content.

9.5 CS 456: Mobile Computing (2)(var)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[X] New Course	[ ] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[ ] New Course Descr	[ ] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Mobile Computing`

CS 456	**Mobile Computing (2)(variable)**
new	Techniques for the routing, broadcasting, and communication
elective	needs in a mobile environment, including issues of database concurrency control, transaction processing, and data consistency. (Prerequisite: CS 301 and IS 351.)

This is a new elective. It is an area of research specialization for Brother Fife.

Please see the next page for a syllabus.

CS 456 Syllabus:

This course will introduce to the student the basic principles and issues of mobile computing. Both traditional and ad-hoc mobile networks will be examined, as well as mobile agents. Issues related to each area will be discussed and include network and database issues.

Possible Text:

- Mobility: Processes, Computers and Agents, D. Milojici, et. al., Addison-Wesley

Topics:

- Traditional Network and Ad-Hoc Networks (1 week)
- Internet Mobility and Mobile agents (3 weeks)
  - Architectures support
  - O.S. support
  - interoperability
  - security and reliability
- Network Mobility (3-5 weeks) (Tradional wireless and Mobile Ad-hoc)
  - connectivity
  - routing
  - Mobile IP
  - power consumption
- Mobile databases (3-4 weeks)
  - Data boradcasting fundamentals
  - data integrity
  - the "broadcast storm"
- Advanced topics - as time permits
  - User identity / user space migration
  - emerging areas of mobile research

9.6 CS 461: Computer Graphics (2)(var)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[ ] New Course Title	[X] Change of Credit Hours	[ ] Change in Fees
[X] New Crs Num	[ ] New Course Descr	[X] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Compu Graphics`

CS 461	**Computer Graphics (2)(variable)**
new	Interactive computer graphics systems programming and
elective	architecture. (Prerequisite: CS 301 and Math 343.)

This is derived from, and replaces:

CS 455	**Computer Graphics (3)(variable)**
old	Interactive computer graphics systems programming and
elective	architecture. (Prerequisite: CS 312, 330 or CS 345, 380.)

The main change is a reduction in credit hours from three to two, with a commensurate reduction in the material covered.

Computer Graphics has always been a very popular course with the students. It is common for male students to get into CS because they played video games and wanted to write such games themselves. Graphics is a key skill in pursuit of that fantasy.

9.7 CS 490R: Advanced Topics in Computer Science (1-3)(var)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[X] New Course	[ ] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[ ] New Course Descr	[ ] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Adv Topics in CS`

CS 490R **Advanced Topics in Computer Science (1-3)(variable)**
Selected advanced topics in Computer Science. (Prerequisite: instructor's consent.)

This course is based upon the old CS 390R course. The only change is the insertion of the word "advanced" into the course title and description.

For accreditation purposes, the CS curriculum must clearly distinguish between courses that are intermediate and those that are advanced. We do this by designating 300-level courses as intermediate, and 400-level courses as advanced.

9.8 CS 491-492-493: Seminar I-II-III (1-1-1)

CS 491- -492-493 **Seminar (1-1-1)(F,W)**
Reading in the Computer Science literature, writing of a review article, research proposal writing and presentation, conduction research, poster presentation, writing and presentation of the senior thesis. (Prerequisite: instructor's consent.)

No changes are proposed for this course. It remains as it was.

Completion of 491-492-493 fulfills the General Education Area III Advanced Writing requirement.

9.9 CS 495R: Independent Study in Computer Science (1-3)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[ ] New Course Descr	[ ] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Indep Study CS`

CS 495R **Independent Study in Computer Science (1-3)(F,W)**
Topic and credit to be arranged between the student and the instructor. (Prerequisite: instructor's consent.)

We add "in Computer Science" to the title. Everything else remains unchanged.

9.10 CS 496R: Student Research in Computer Science (1-3)

NATURE OF THE CHANGE: {Check the appropriate box(es)}

[ ] New Course	[X] New Course Title	[ ] Change of Credit Hours	[ ] Change in Fees
[ ] New Crs Num	[ ] New Course Descr	[ ] Add/change Prerequisite(s)	[ ] Other

Abbreviated Title (16 char): `Stu Research CS`

CS 496R **Student Research in Computer Science (1-3)(F,W)**
Supervised individual research for students who have been granted a student research and development assistantship. (Required for all associates.)

We add "in Computer Science" to the title. Everything else remains unchanged.