CS 135 Discrete Structures
Spring 2014

Catalog description:
The aim of this course is to integrate knowledge of basic mathematics with the problems involving specification, design, and computation. By the end of the course, the student should be able to: use sets, functions, lists, and relations in the specification and design of problems; use properties of arithmetic, modular arithmetic (sum, product, exponentiation), prime numbers, greatest common divisor, factoring, Fermat's little theorem; use binary, decimal, and base-b notation systems and translation methods; use induction to design and verify recursive programs; and implement in Scheme all algorithms considered during the course.

Instructor:

Matt Burlick (mburlick@stevens.edu)
Office Hours: Lieb Building, Room 214
W 1:00-3:00pm
R 2:00-4:00pm
And by appointment (subject to change)

Teaching Assistant:

Sadia Akhter (sakhter@stevens.edu)
Dominik Jedruszczak (djedrusz@stevens.edu)

Graders:

Sadia Akhter (sakhter@stevens.edu)
Dominik Jedruszczak (djedrusz@stevens.edu)
Emre Karabacak (ekarabac@stevens.edu)
Gaurav Sharma (gsharma1@stevens.edu)

Meetings:

Lecture/Recitation:	MWF	10:00am – 10:50am	Stevens Hall 230
Lecture/Recitation:	W	10:00am – 10:50am	Babbio 122
Lecture/Recitation:	F	10:00am – 10:50am	Burchard 118

Labs:

Lab A	T	9:00am - 10:40am	Babbio 304
Lab B	T	11:00am - 12:40pm	Babbio 310
Lab C	T	3:00pm - 4:40pm	Morton 103

Prerequisite:
There is no course prerequisite, but some programming experience will be helpful.

Required textbooks:
Discrete Mathematics and Its Applications, 7th edition 2012, by Kenneth Rosen, ISBN: 0073383090. You must have the 7th Edition!

The Little Schemer - 4th Edition, by Daniel P. Friedman and Matthias Felleisen, ISBN 0-262-56099-2

Software:
- DrRacket (a programming environment for the language also known as Scheme)

Policies and Grading

- You, your instructor, and the TA are bound by the Stevens Honor Code. Students are responsible for reading and understanding the course policies in this syllabus and for announcements made in class and in the course email list.

- You will be permitted to use the textbooks and course notes for programming assignments (homework and labs). During exams, you are **not** permitted to use notes, books, computing or communication devices unless a different policy is specifically announced by the instructor.

- Notebook computers are not to be used in class unless otherwise specified.

- During lecture and lab sessions please refrain from using mobile phones or otherwise being impolite.

- The course score is a weighted average of the following categories.

Lab	20%
Homework	20% (including pop quizzes)
First exam	15%
Second exam	20%
Final exam	25%

- Labs will be graded as follows:

Completion	100%
Partial Completion	75%
No attempt	25%
Absence	0%

- The course score is on a scale of 100 and letter grades (including plusses and minuses). In addition to the grade break-up provided above, effort and progress may be taken into account when computing your final grade. Final letter grades will be assigned according to class-wide grade clustering.

- Attendance will be taken at both lectures and labs (attendance is mandatory). You are permitted 1 missed lab.

- The instructor reserves the right to give a higher grade than your course score, if your performance on later assignments and exams is very strong.

- Labs are "closed": the assignment is given, completed, and graded during the lab session. Group work is at the discretion of the TA.

- There are no make-ups for labs or exams. The only possible exceptions are in the case of death in the student's immediate family or near-death experience of the student; advance notice is required.

- There may be short, unannounced quizzes in class, which count in the "homework" grade category. The purpose is to motivate attendance and to help both you and your instructor gauge your progress.

- This is a course on discrete math, not on Scheme. We will use a study a small, subset of the language in order to (a) develop skill in pure functional programming and (b) focus on the math. There will be about 10 homework assignments, often involving small Scheme programs.

- Homework will be accepted up to 48 hours late: up to 24 hours late will get a penalty of 20% grade reduction; up to 48 hours late gets penalty of 50%.

- Except when groups are explicitly allowed, work must be done individually. You are encouraged to discuss the problems with your classmates but you must not share details of the solutions. If you are unsure whether you have shared too much, discuss the situation with the TA or instructor; it is your obligation to avoid even the appearance of cheating.

CS 135 Week by Week

The following is an outline for the course.
Sections refer to the Rosen textbook, 7th edition, unless marked LS which means The Little Schemer.

Week	**Topic(s)**	**Reading**
Week 1	Intro to the Course and to Logic	Sections 1.1-1.3
Week 2	Predicate Logic	Section 1.4
Week 3	Intro to Proofs, Set Theory	Sections 1.6-1.8
Week 4	Sets, Functions, Sequences	Sections 2.1-2.5, LS Chapters 1-2
Week 5	Recursion	Section 5.4
Week 6	Mathematical Induction	Section 5.1
Week 7	Structural Induction	Section 5.3
Week 8	Review and Exam	
Week 9	Tail Recursion	
Week 10	Relations	Sections 9.1-9.3
Week 11	Closures, Equivalence	Section 9.4
Week 12	Integers, Division	Section 4.1
Week 13	Number Theory	Sections 4.2-4.3
Week 14	Cryptography, Graphs	Section 4.6
Week 15	Graphs and Trees	Sections 10.1-10.3, 11.1-11.3

CS 135 Goals and assessment

To assess student progress we focus on key skills that can be demonstrated. Here is the official list of course outcomes to be achieved by the end of the semester:

Formal logic - Unfold a definition, apply a theorem, use deductive rules, for example in proving that a given relation is a function or a function is injective.

Relations - Define basic properties of relations such as transitivity and injectivity, in words and using logical formulas.

Scheme - Implement recursive functions on lists, such as append or reverse. For relations represented by lists of pairs, implement operations such as transposition and intersection.

Induction - Use induction and equational reasoning to prove equations about recursive functions, such as associativity of append.

Modular arithmetic - Simplify expressions by using laws of modular arithmetic.

Primes - Define basic properties such as relative primality. Implement Euclid's algorithm in Scheme.