COURSE DESCRIPTION

Dept., Number	**CSC 28**	Course Title	**Discrete Structures for Computer Science**
Semester hours	**3**	Course Coordinator	**Ted Krovetz**
		URL (if any):	**http://krovetz.net/csus/**

Catalog Description

Introduction to the essential discrete structures used in computer science, with emphasis on their applications. Topics include: counting methods, elementary formal logic and set theory, recursive programming, digital logic and combinatorial circuits, real number representation, regular expressions, finite automata. Prerequisite: Math 29 and CSc 20; CSc 20 may be taken concurrently.

Textbook

Seymour Lipschutz and Mark Lipson, Shaum's Outline of Discrete Mathematics, 3rd Ed., McGraw-Hill, 2007.

Daniel J. Velleman, How to Prove it: A Structured Approach, 2nd Edition, Cambridge University Press, 2006.

Course Goals

1. To provide basic and theoretical competencies that are explicitly used in upper-division Computer Science core courses.
2. To help students understand and appreciate the basic mathematical knowledge which is fundamental to Computer Science.
3. To introduce aspects of computer theory and computer organization that rely directly on that knowledge.

Prerequisites by Topic

Basic understanding of:

- Programming, including loops and one-and two-dimensional arrays.
- Pre-calculus mathematics.

Exposure to:

- Recursion.

Major Topics Covered in the Course

1. Counting methods: basic principles, permutations and combinations, the pigeonhole principle (4.5 hours).
2. Sets and relations: sets, sequences and strings, relations, functions, review of basic functions, including modular arithmetic (7 hours).

3. Common numerical sets, representation of real numbers (bias, mantissa and negative numbers) and related limitations (1.5 hours).
4. Logic and proofs: propositions, conditional propositions and logical equivalence, quantifiers, proofs, mathematical induction (6 hours).
5. Recursive programs: problem solving with well-structured recursive functions, recursive procedures, implementation and efficiency of recursive algorithms (6 hours).
6. Digital logic: Boolean algebra, logic gates and combinatorial circuits, circuit design methodology, reduction to nor/nand gates, circuit minimization, Karnaugh maps including don't care (10.5 hours).
7. Languages: generation vs. recognition, regular expressions, finite-state machines (7.5 hours).
8. Additional topics (e.g. asymptotics, relational algebra (2 hours).

Outcomes

Thorough understanding of:

- Sets and basic operations on sets.
- Method used in the construction of a recursive algorithm to solve a word problem.
- Conversion of an English-language statement of compound propositions (including and, or, not, and quantifiers) to a symbolic form.
- Determination of the logical equivalence of propositions and the validity of formal arguments via truth tables.

Basic understanding of:

- Proof methods including proof by induction and by contradiction.
- Relations and functions, their definition and properties.
- Basic arithmetic functions: abs, ceiling, floor, exponential, logarithm, integer division, modular arithmetic.
- Design and construction of a combinatorial circuit from a verbal description.
- Regular expressions (can compose a valid expression for a simple regular language).
- Finite automata (is able to construct a valid recognizer for a simple regular language).

Exposure to:

- Basic counting principles.
- IEEE floating-point standard.

Laboratory Projects

Assignments designed to assist the students to understand the fundamental theoretical principles are assigned as homework and also done in a group as part of the lectures.

Estimated Curriculum Category Content (Semester hours)

Area	*Core*	*Advanced*	*Area*	*Core*	*Advanced*
Algorithms	0.5		Data Structures		
Software Design			Prog. Languages		
Comp. Arch.	1.0				

Oral and Written Communications

No significant component.

Social and Ethical Issues

No significant component.

Theoretical Content

Two-thirds of the course is theoretical.

Problem Analysis

No significant component.

Solution Design

No significant component.

/sj