UNIVERSITY OF NEBRASKA AT OMAHA
COURSE SYLLABUS

Department and Course Number	Computer Science, CSCI 2030
Course Title	Mathematical Foundations of Computer Science
Course Coordinator	Mahadevan Subramaniam
Total Credits	3
Repeat for Credit?	Yes
Date of Last Revision	April 21, 2011.

1.0 Course Description Information

1.1 Catalog description: This course introduces discrete mathematics concepts that are foundational for the study of computer science. The covered topics include functions, relations, and sets, basic logic, methods of proof, mathematical induction, computational complexity, recursion, counting, recurrences, and relations.

Prerequisites of the course: MATH 1950 or MATH 1930

1.2 Overview of content and purpose of the course: This course introduces discrete mathematics concepts that are foundational for the study of computer science. The covered topics functions, relations, and sets, basic logic, methods of proof, mathematical induction, computational complexity, recursion, counting, recurrences, and relations. The purpose of this course is to impart to students the knowledge to describe and analyze computational problems in a logical and rigorous way.

1.3 Unusual circumstances of the course.

None.

2.0 Course Justification Information

2.1 Anticipated audience / demand:

The course is intended for computer science students with sophomore standing.

2.2 Indicate how often this course will be offered and the anticipated enrollment:

It will be offered every semester with an anticipated enrollment around 40-70 students.

2.3 If it is a significant change to an existing course, please explain why it is needed:

This is a new course.

3.0 List of performance objectives stated in learning outcomes in a student's perspective:

3.1 This course seeks to foster the understanding and use of discrete structures that are backbone of computer science with an emphasis on the applications for students majoring in computer science. On successful completion of this course, a student will be:

3.1.1 Proficient in the logical concepts of propositional and aspects of first-order logic including quantification and proof methods based on these logical foundations.

3.1.2 Able to comprehend and construct proofs used to reason about the working and correctness of several computing artifacts including data types, algorithms, and programs.

3.1.3 Proficient in the use of recursive methods for constructing finite and infinite sets, functions, and data types and understand the close relationship between inductive reasoning and recursive specifications.

3.1.4 Able to comprehend and formulate recursive sets and functions, and data types and reason about the behavior of these recursive structures using mathematical and structural induction methods.

3.1.5 Proficient in basic counting principles including inclusion-exclusion, pigeon-hole principle, binomial theorem, and recurrences.

3.1.6 Able to use counting techniques to quantitatively estimate the space and the time requirements of data, and algorithms designed to solve problems in several computer applications.

3.1.7 Students will also learn the theory of relations and be able to reason about relations specified using computer databases.

4.0 Content and Organization Information

4.1 The Foundations: Logic and Proofs (8 hours)
- 4.1.1 Propositional Logic
- 4.1.2 Propositional Equivalences
- 4.1.3 Predicates and Quantifiers
- 4.1.4 Nested Quantifiers
- 4.1.5 Rules of Inference
- 4.1.6 Introduction to Proofs
- 4.1.7 Proof Methods and Strategy

4.2 Basic Structures: Sets, Functions, Sequences, and Sums (4 hours)
- 4.2.1 Sets
- 4.2.2 Set Operations
- 4.2.3 Functions
- 4.2.4 Sequences and Summations

4.3 The Fundamentals: Algorithms, the Integers, and Matrices (3 hours)
- 4.3.1 Algorithms
- 4.3.2 The Growth of Functions
- 4.3.3 Complexity of Algorithms

4.4 Induction and Recursion (6 hours)
- 4.4.1 Mathematical Induction
- 4.4.2 Strong Induction and Well Ordering
- 4.4.3 Recursive Definitions and Structural Induction
- 4.4.4 Recursive Algorithms

4.5 Counting (4 hours)
- 4.5.1 The Basics of Counting

4.5.2 The Pigeonhole Principle
4.5.3 Permutations and Combinations
4.5.4 Generating Permutations and Combinations

4.6 Advanced Counting Techniques (3 hours)
4.6.1 Recurrence Relations

4.7 Relations (6 hours)
4.7.1 Relations and Their Properties
4.7.2 Representing Relations
4.7.3 Equivalence Relations
4.7.4 Partial Orderings

4.8 List the major topics central to this course:

5.0 Teaching Methodology Information

5.1 Methods:

The primary teaching methods will be lecture, in-class demonstrations, and in-class exercises, in class quizzes, homework assignments, and examinations.

5.2 Student role:

The student will attend lectures and demonstration, participate in discussion on assigned readings, complete assigned homework, and complete required examinations.

6.0 Evaluation Information

Describe the typical types of student projects that will be the basis for evaluating student performance:

Students are asked to complete assignments intended to enhance theoretical analysis and proofs. This is in addition to exams given in the course. There will be no student projects.

6.1 Describe the typical basis for determining the final grade (e.g. weighting of various student projects):

Component	Grading
Exams	55%
Assignments and Quizzes	40%
Participation	5%

6.2 Grading type:

Points	Grade
97-100%	A+
93-96%	A
90-92%	A–
87-89%	B+
83-86%	B
80-82%	B–
77-79%	C+
73-76%	C
70-72%	C–
67-69%	D+
63-66%	D
60-62%	D–
0-59%	F

7.0 Resource Material Information

7.1 Textbooks and/or other required readings used in course:

K. H. Rosen, *Discrete Mathematics and Its Applications*, 6th Edition, McGraw Hill, 2007.

7.2 Other student suggested reading materials:

1) E. Lehman, and T. Leighton, *Mathematics for Computer Science*, 2004, http://www.cs.princeton.edu/courses/archive/fall06/cos341/handouts/mathcs.pdf

2) L. Liu, *Elements of Discrete Mathematics*, 2nd Edition, McGraw Hill, 1985.

3) G. Pol`a, How to Solve it: *A New Aspect of Mathematical Method*, Ishi Press, 2009.

7.3 Current bibliography and other resources:

- A. Benjamin, and J.J. Quine, *Proofs that really count*, Mathematical Association of America, 2003.
- R.L. Graham, D.E. Knuth, O. Patashnik, *Concrete Mathematics*, 2nd Edition, Addison-Wesley, 1994.
- D. Gries and F. B. Schneider, *A logical approach to Discrete Math*, Springer, 1993.
- L. Lovasz, J. Pelikan, and K.L. Vesztergombi, *Discrete Mathematics*, Springer 2003.
- E.S. Roberts, *Thinking Recursively*, Wiley, 1987.
- K.H. Rosen, Handbook of Discrete and Combinatorial Mathematics, CRC Press, 2000.

- D, Solow, *How to read and Do Proofs: An Introduction to Mathematical Thought processes*, Wiley 2001.
- R. Smullyan, *The Riddle of Schehrazade: And Other Amazing Puzzles, Ancient and Modern*, Harvest Books, 1998.
- A. Tucker, *Applied Combinatorics*, 4th Edition, Wiley, 2002.
- D. Velleman, *How to Prove It: A Structured Approach*, Cambridge University Press, 1994.

8.0 Other Information:

8.1 Accommodations statement:

8.2 Other:

8.3 Author(s):

9.0 Computer Science Accreditation Board (CSAB) Category Content (class time in hours):

CSAB Category	*Core*	*Advanced*
Data structures		
Computer organization and architecture		
Algorithms and software design		
Concepts of programming languages		

10.0 Oral and Written Communications:

Every student is required to submit at least 0 written reports (not including exams, tests, quizzes, or commented programs) to typically 0 pages and to make 0 oral presentations of typically 0 minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

11.0 Social and Ethical Issues: No coverage.

12.0 Theoretical content: Discrete mathematics is well-founded in the theoretical aspects of computer science.

13.0 Problem analysis: The course is mathematical in nature and, as a result, requires considerable understanding of the methods used to solve mathematical problems in the course. This involves problem detailed problem reading to understand what is given and what is required. This is followed by analysis applied to special cases for better understanding of the problem requirements.

14.0 Solution design: The solution design includes verification and proof of correctness. In particular, proof by contradiction, and inductive proofs are used extensively. The design involves the implementation of software objects to encapsulate the discrete structures that the students use to solve the problems.

CHANGE HISTORY

Date	***Change***	***By whom***	***Comments***
03/11/2010	Initial Version	Jon Yoon	
04/02/2010	Minor revision of Topics	Jon Youn	
04/21/2011	Revision of Topics Change learning outcomes Change bibliography	Mahadevan Subramaniam	Topics were revised based on the text used in Spring 2011 and discussions within UPC and faculty teaching the course. Learning outcomes and bibliography were changed to new format and to address VC comments.

S – Strong relationship
X – Contributing relationship

Course objective	**(a) knowledge of discipline**	**(b) analyze problem, define requirements**	**(c) design and implement solution**	**(d) function on a team**	**(e) ethical issues**	**(f) communicate effectively**	**(g) analyze impact of computing**	**(h) continued professioanl development**	**(i) Current techniques and tools**	**(j) apply foundations**	**(k) apply design and development principles**
CSCI 2030	**S**	**S**	**C**	**C**	**C**	**C**	**S**	**C**	**C**	**S**	**C**