

CSC 333 - DISCRETE STRUCTURES FOR COMPUTER SCIENCE

CREDIT HOURS: 3

PREREQUISITES: CSC 202; MTH 233 or 144

GRADE REMINDER: Must have a grade of C or better in each prerequisite course.

CATALOG DESCRIPTION

Mathematical structures for describing data, algorithms, and computing machines. Theory and application of sets, relations, functions, combinatorics, matrices, graphs, and algebraic structures which are pertinent to computer science.

PURPOSE OF COURSE

To develop logical and mathematical concepts necessary to understand and analyze computational systems. Introduce concepts, techniques, and skills necessary to comprehend the underlying structure of problems encountered in designing and implementing computer systems and software. Provide the foundations for understanding computer science topics that rely upon the comprehension of formal abstract concepts.

EDUCATIONAL OBJECTIVES

Upon successful completion of the course, students should be able to:

1. Use formal notation for propositional and predicate logic.
2. Construct formal proofs in propositional and predicate logic and use such proofs to determine the validity of English language arguments.
3. Prove conjectures using the techniques of direct proof, proof by contraposition, proof by contradiction, and proof by induction.
4. Prove the correctness of programs that contain looping constructs.
5. Demonstrate an understanding of recursive definitions and to write recursive definitions for certain sequences and collections of objects.
6. Describe how recursive algorithms execute.
7. Use set notation and set operations to prove/disprove set identities.
8. Use the Principle of Inclusion and Exclusion to find the number of elements in the union of sets.
9. Solve permutation and combination problems for a set of n distinct objects.
10. Use relations and functions and apply these concepts to ordering problems.
11. Use graphs, directed graphs, and trees as representation tools in a wide variety of contexts.

CONTENT

Hours

Introduction to Formal Logic with applications:	6
Logic as a model for computation e.g. the Prolog programming language	
Using logic as a tool to design computer components	
Logical expressions and circuits	
Design and simplification of logical expressions using Karnaugh maps	
Implicants and coverings of Karnaugh maps truth tables, tautologies, and the evaluation of conditional expressions in programming languages	
The satisfiability problem and NP-completeness	
Truth and provability	
Inference using resolution	
Introduction to Proofs and Recurrence Relations with applications:	8
Repetition, recursion, induction in programs	
Use of inductive definitions for data models	
Use of inductive proofs for describing error-detecting codes	
Proofs of program correctness	
Recursive definitions, e.g. parenthesized expressions	
Analysis of algorithms	
Running time of programs with iterative structures, programs with procedure calls, programs with recursive procedure calls, conditional statements	
Use of recurrence relations to describe running time of programs	
Solving recurrence relations	
Proving properties of programs	
Properties of programs that cannot be proved (unsolvable problems)	
Sets and Combinatorics with applications:	8
The set data model	
Operations on sets: insert, delete, lookup	
Enumerative algorithms and complexity classes	
Algorithms for union, intersection, and difference	
Relations, Functions, and Matrices with applications:	8
Operations on functions: insert, delete, lookup	
Partial orders and task system precedence constraints	
Total orders and execution sequencing	
Transitivity (pitfalls), symmetry, computing closures of relations	
The relational data model, keys and indexes, operations	
Projection and join operations	
Matrices as a data structure for representing relations	
Graphs and Trees with applications:	8
Representations of graphs using lists and matrices	
Paths and circuits in graphs	
Algorithms for detecting connected components	
Warshall's algorithm for reachability and transitive closure in a network	
Depth-first search, network broadcast patterns	

Planarity in circuit layout
Use of cycle detection in file systems and concurrent systems
Applications of graph coloring: cliques, conflict scheduling

Introduction to Algebraic Structures, Languages, and Machines with applications: 4
 Encoding and decoding
 Pattern specification and matching
 Properties of regular expressions, UNIX specification of patterns
 Equivalence of regular expressions and finite state automata
 Minimization of automata
 Graphs to represent state machines
 Automata and their programs
 Nondeterminism

Exams 3

TOTAL 45

REFERENCES

Aho, A. and Ullman, J., Foundations of Computer Science, C Edition, W.H. Freeman, 1995.

Doerr, A. and Levasseur, K., Applied Discrete Structures for Computer Science, SRA, 1985.

Gersting, J., Mathematical Structures for Computer Science, 5th Ed., W.H. Freeman, 2003.

Grimaldi, R., Discrete and Combinatorial Mathematics, 3rd Edition, Addison-Wesley, 1994.

Hein, J.L., Discrete Structures, Logic, and Computability, Jones and Bartlett, 1995.