COURSE OUTLINE

COS204 — Course Number

Discrete Mathematical Structures — Course Title

4 — Credits

4 Hours Lecture/Week — Hours: lecture/laboratory/other

Catalog description:

Discrete mathematics is primarily intended for computer science majors. The course covers a wide variety of topics serving as the mathematical framework for the design and analysis of algorithms. Topics include induction and recursion, relations, functions, sets, propositional logic, Boolean algebra, grammars, tree structures, permutations and combinations and finite state machines.

Prerequisites: MAT151 (Calculus I)

Corequisites: NONE

Required texts/other materials:

Kenneth H Rosen, Discrete Mathematics and Its Applications, McGraw-Hill Education

Last revised: Spring 2019

Course coordinator: Meimei Gao

Information resources:

Other learning resources:

Course goals:

The student will be able to:

- describe and integrate basic definitions and theorems concerning sets, functions, and relations
- use mathematical tools of logic and induction and show the application of these tools to computer science
- create and understand a formal proof
- examine formal languages and methods for representing grammars
- use combinations and probability theory required in the design and analysis of algorithms
- create state and transition diagrams to describe a system

General Education goals and objectives.

Critical thinking, problem solving and information literacy: Students will use critical thinking and problem-solving skills in analyzing information gathered through different media and from a variety of sources.

Students will identify a problem and analyze it in terms of its significant parts and the information needed to solve it.
Students will use computers to access, analyze or present information, solve problems, and communicate with others.
Students will formulate and evaluate possible solutions to problems, and select and defend the chosen solutions.
Students will recognize weaknesses in arguments, such as the use of false or disputable premises, suppression of contrary evidence, faulty reasoning, and emotional loading.

Quantitative skills: Students will apply appropriate mathematical and statistical concepts and operations to interpret data and to solve problems.

Students will translate quantifiable problems into mathematical terms and solve these problems using mathematical or statistical operations.
Students will construct graphs and charts, interpret them, and draw appropriate conclusions.

Unit 1. Fundamentals

At the completion of Unit 1, the student should be able to:

1. Appreciate how the concept of a set is fundamental to both mathematics and computer science.
2. Perform the common operations on sets such as union, intersection, and difference.
3. Visualize set operations in terms of Venn diagrams.
4. Use algebraic laws to manipulate and simplify expressions involving sets and operations on sets.
5. Represent sets using characteristic vectors.
6. Define a sequence with a recursive or explicit formula.
7. Compute the regular set corresponding to a regular expression.
8. Describe recursive algorithms such as factorial, Euclid's algorithm, and the Fibonacci sequence.
9. Compute the sum, product, and transpose of matrices.
10. Compute the join, meet, and product of Boolean matrices.

Unit 2. Logic

At the completion of Unit 2, the student should be able to:

1. Apply formal methods of symbolic propositional and predicate logic.
2. Apply the rules and techniques needed for determining whether a given argument is valid.
3. Use propositional logic to prove theorems.
4. Create truth tables to represent logical expressions.
5. Explain common proof techniques such as "proof by contradiction" and "proof by contrapositive."
6. Use deduction and induction as methods of proof.
7. Use induction to verify program correctness.

Unit 3. Proof Techniques

At the completion of Unit 3, the student should be able to:

1. Outline the basic structure of and give examples of each proof technique.
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.
5. Apply proof techniques to solve problems in computer science, including software engineering, program semantics, and algorithm analysis.

Unit 4. Counting

At the completion of Unit 4, the student should be able to:
1. Determine the existence of a solution to a problem.
2. Use combinatorial analysis techniques to count the number of solutions.
3. Determine the optimization algorithm.
4. Use the multiplication principle of counting.
5. Compute the number of permutations of a set of objects.
6. Compute the numbers of combinations of a set of objects.
7. Apply the Pigeonhole Principle.
8. Analyze probabilistic experiments.
9. Find an explicit formula for a recursively defined sequence.

Unit 5. Relations and Digraphs

At the completion of Unit 5, the student should be able to:
1. Apply the concept of a binary relation.
2. Represent relations with matrices and digraphs.
3. List ordered pairs in a Cartesian product.
4. Obtain a partition of a set.
5. Apply the reachability relation.
6. Define reflexive, irreflexive, symmetric, asymmetric, antisymmetric and transitive relations.
7. Use adjacency matrices and digraphs to represent these relations.
8. Define an equivalence relation.
9. Implement relations using linked lists and arrays.
10. Define the composition of two relations.
11. Apply Warshall's algorithm to compute transitive closure.

Unit 6. Functions
Order Relations

At the completion of Unit 6, the student should be able to:
1. Define the concept of a function as a type of relation.
2. Define one-to-one, everywhere defined, and onto functions.
3. Compute inverse functions.
4. Analyze bijections, transpositions, and cyclic permutations.
5. Define a partially ordered set, linearly ordered set, and lexicographic ordering.
6. Diagram posets using Hasse diagrams.
7. Construct a linear order using topological sorting techniques.

Unit 7. Trees
Finite-State Machines

At the completion of Unit 7, the student should be able to

1. Appreciate how the concept of a tree is exceptionally useful in computer science.
2. Use the terms root, vertex, level, parent, offspring, siblings, height and leaves in discussing trees.
3. Draw digraphs to represent ordered or labeled trees.
4. Traverse trees with preorder, inorder, and postorder algorithms.
5. Define a phrase structure grammar and construct derivation trees.
6. Represent finite-state machines with state transition tables and digraphs.
7. Draw the digraph of a Moore machine and construct a quotient Moore machine.
8. Understand the connection between the language of a machine and phrase structure grammar.

Evaluation of student learning:

To ensure academic freedom for the individual faculty member the following is only suggested:

3 tests	60%
final exam	40%
	100%

Academic Integrity Statement:

Mercer County Community College is committed to Academic Integrity -- the honest, fair and continuing pursuit of knowledge, free from fraud or deception. This implies that students are expected to be responsible for their own work and that faculty and academic support services staff members will take reasonable precautions to prevent the opportunity for academic dishonesty.

See http://www.mccc.edu/admissions_policies_integrity.shtml for a complete explanation of policies and procedures regarding academic integrity and academic integrity violations.