Miami Dade College

**Course Description**

**COT3100C | Discrete Structures | 4.00 credits**

This course delves into discrete structures with a strong focus on applications in computer science. It covers a range of topics including sets, functions and relations, propositional and predicate logic, basic proof techniques, logic, recursion and induction, graphs, Big-O notation, and various specific applications relevant to computer science. Prerequisite: COP2800; Corequisite: COP3530.

**Course Competencies:**

**Competency 1:** The student will demonstrate knowledge of set functions and relations as they apply to computer science by:

- Performing set operations (union, intersection, difference, complement) and proving properties about sets (e.g., proving that the union of two sets is commutative)
- Analyzing functions, identifying their properties (injective, surjective, bijective), finding inverses, composing functions, and applying them to computational problems
- Identifying properties of relations (reflexivity, symmetry, transitivity, antisymmetry) and working with equivalent relations and partial orders

**Competency 2:** The student will demonstrate knowledge of number theory by:

- Performing calculations within modular systems, understanding congruences, and applying them to solve equations or prove properties
- Finding the greatest common divisor (gcd) or least common multiple (lcm) of integers using algorithms like the Euclidean algorithm, with applications in computational efficiency and algorithms
- Using primality testing algorithms (e.g., primality by divisibility, Fermat's Little Theorem) to determine if a number is prime or composite

**Competency 3:** The student will demonstrate knowledge of counting as it applies to computer science by:

- Solving problems that involve counting the number of ways to arrange or select items, distinguishing between cases where order does or does not matter
- Applying the binomial theorem and using Pascal's triangle to solve counting problems, including finding specific coefficients in binomial expansions
- Calculating the number of ways to select a group of objects without regard to order
- Applying the inclusion-exclusion principle to count objects that satisfy certain conditions, with applications in computational counting problems

**Competency 4:** The student will demonstrate knowledge of logic by:

- Constructing Truth Tables to determine the truth values of compound propositions and to evaluate logical equivalences
- Solving problems that require the application of logical equivalences to simplify logical expressions or to prove statements about them
- Solving problems involving quantifiers (existential and universal), interpreting and constructing predicate logic expressions, and understanding their scope and implications

**Competency 5:** The student will demonstrate knowledge of induction as it relates to computer science by:

- Solving problems that require direct proof, where they use definitions, theorems, and logical deductions to prove statements related to algorithm correctness
- Solving proof by contradiction, including problems that are best or only provable by assuming the negation of the statement to be proven leads to a contradiction

- Using mathematical induction to prove statements about integers, including the base case, induction hypothesis, and induction step, with applications in algorithm correctness and other computational processes

**Competency 6:** The student will demonstrate knowledge of graphs by:

- Defining basic graph terminology like vertex, edge, directed vs. Undirected graphs, weighted vs. Unweighted graphs, and different graph types (complete graphs, bipartite graphs, etc.)
- Solving problems that involve representing graphs using adjacent matrices, adjacency lists, and incidence matrices
- Identifying properties of graphs, such as connectivity, cycles, paths, degrees of vertices, and whether a graph is Eulerian or Hamiltonian, with applications in computing

**Competency 7:** The student will demonstrate knowledge of recurrence relations by:

- Solving linear homogeneous and non-homogeneous recurrence relations, possibly with constant coefficients, using methods like iteration, the characteristic equation, or generating functions
- Solving the recurrence relation by expressing subsequent terms based on the previous term(s) and the base case(s), using the back substitution method
- Solving recurrence relations using the master theorem for reducing and dividing functions
- Solving recurrence relations using the tree method