GROSSMONT COLLEGE

 Official Course Outline

COMPUTER SCIENCE INFORMATION SYSTEMS 240 – DISCRETE STRUCTURES

 1. Course Number Course Title Semester Units Semester Hours

 CSIS 240 Discrete Structures 3 3 hours lecture: 48-54 hours

 96-108 outside-of-class hours

 144-162 total hours

 2. Prerequisites

A “C” grade or higher or “Pass” in CSIS 293 or equivalent.

Corequisite

 None.

 Recommended Preparation

 None.

 3. Catalog Description

This course is an introduction to the discrete structures used in computer science with an emphasis on their applications. Topics covered include: Functions, Relations and Sets; Basic Logic; Proof Techniques; Basics of Counting; Graphs and Trees; and Discrete Probability.

 4. Course Objectives

 The student will:

1. Describe how formal tools of symbolic logic are used to model real-life situations, including those arising in computing contexts such as program correctness, database queries, and algorithms.
2. Relate the ideas of mathematical induction to recursion and recursively defined structures.
3. Analyze a problem to create relevant recurrence equations.
4. Demonstrate different traversal methods for trees and graphs.
5. Apply the binomial theorem to independent events and Bayes’ theorem to dependent events.

 5. Instructional Facilities

 Standard computer lab with one internet-connected workstation per student with appropriate software installed.

6. Special Materials Required of Student

Flash/USB drive or cloud storage for backup of in-class work.

7. Course Content

1. Functions (surjections, injections, inverses, composition)
2. Relations (reflexivity, symmetry, transitivity, equivalence relations)
3. Sets (Venn diagrams, complements, Cartesian products, power sets)
4. Pigeonhole principles
5. Cardinality and countability
6. Basic logic, propositional logic, and logical connectives
7. Truth tables
8. Normal forms (conjunctive and disjunctive)
9. Validity

COMPUTER SCIENCE INFORMATION SYSTEMS 240 – DISCRETE STRUCTURES Page 2

7. Course Content (continued)

1. Predicate logic
2. Universal and existential quantification
3. Modus ponens and modus tollens
4. Limitations of predicate logic
5. Proof techniques
6. Notions of implication, converse, inverse, contrapositive, negation, and contradiction
7. The structure of mathematical proofs, direct proofs, proof by counterexample, and proof by contradiction
8. Mathematical induction and strong induction
9. Recursive mathematical definitions
10. Well orderings
11. Basics of counting, counting arguments, and sum and product rule
12. Inclusion-exclusion principle
13. Arithmetic and geometric progressions
14. Fibonacci numbers
15. Permutations and combinations
16. Basic definitions
17. Pascal’s identity
18. The binomial theorem
19. Solving recurrence relations
20. The master theorem
21. Graphs and Trees:

Trees, Undirected graphs, Directed graphs, Spanning trees/forests, Traversal strategies

1. Discrete Probability:

 (1) Finite probability space, probability measure, events

 (2) Conditional probability, Independence, Bayes’ theorem

 (3) Integer random variables, expectation

 (4) Law of large numbers

 8. Method of Instruction

1. Lecture
2. Demonstration such as recreating a simple DAG (Directed Acyclic Graph) of DLT (Distributed Ledger Technology) transactions from the IOTA network on a white board along with seeing the online visual animation of real time transactions. Another example would be to use string to connect various standing students around the room where the students represent vertices and the string represents edges in the simulation of a graph structure while hanging taped names on all the edges.
3. Student exercises
4. Reading assignments

 9. Methods of Evaluating Student Performance

1. Classroom exercises such as describe in writing how Venn Diagrams can be used to demonstrate De Morgan's Laws.
2. Project activities such as write a Java program that implements Prim's algorithm for generating a minimum spanning tree.

 c. Objective examinations and quizzes including a final examination.

10. Outside Class Assignments

 a. Discrete Structure algorithm and problem-solving exercises.

 b. Textbook reading assignments.

 c. Projects such as write a Java program that implements Prim's algorithm for generating a minimum spanning tree and allows a user to enter node values contained within a student-chosen domain such as local towns and the distances separating them.

COMPUTER SCIENCE INFORMATION SYSTEMS 240 – DISCRETE STRUCTURES Page 3

11. Texts

 Required Text(s):

 Kolman, Bernard and Robert Busby. *Discrete Mathematical Structures*. 6th edition, Pearson, Boston, MA, 2017.

 Addendum: Student Learning Outcomes

 Upon completion of this course, our students will be able to do the following:

1. Describe how formal tools of symbolic logic are used to model real-life situations, including those arising in computing contexts such as program correctness, database queries, and algorithms.
2. Relate the ideas of mathematical induction to recursion and recursively defined structures.
3. Analyze a problem to create relevant recurrence equations.
4. Demonstrate different traversal methods for trees and graphs.
5. Apply the binomial theorem to independent events and Bayes’ theorem to dependent events.

Date approved by the Governing Board: December 11, 2018