Lecture Contact Hours: 48-54, Homework Hours: 96-108, Total Student Learning Hours: 144-162

CUYAMACA COLLEGE
COURSE OUTLINE OF RECORD

COMPUTER SCIENCE 240 – DISCRETE STRUCTURES

3 hours lecture, 3 units

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.

Prerequisite
"C" grade or higher or "Pass" in CS 181 or CS 182 or equivalent, or experience programming in C/C++ or Java

Entrance Skills
Without the following skills, competencies and/or knowledge, students entering this course will be highly unlikely to succeed:
1) Identify common tools, functions, and utilities available in modern programming languages.
2) Define and utilize structured programming principles.
3) Utilize the three basic logic control structures of sequence, decision and repetition.
4) Understand arrays.
5) Trace the flow of execution and values of all variables through a simple program.

Course Content
1) Programming with high level languages like C++, Java, and Python
2) Functions (surjections, injections, inverses, composition)
3) Relations (reflexivity, symmetry, transitivity, equivalence relations)
4) Sets (Venn diagrams, complements, Cartesian products, power sets)
5) Pigeonhole principles
6) Cardinality and countability
7) Basic Logic
8) Propositional logic
9) Logical connectives
10) Truth tables
11) Normal forms (conjunctive and disjunctive)
12) Validity Predicate logic
13) Universal and existential quantification
14) Modus ponens and modus tollens
15) Limitations of predicate logic
16) Proof Techniques
17) Notions of implication, converse, inverse, contrapositive, negation, and contradiction
18) The structure of mathematical proofs
19) Direct proofs
20) Proof by counterexample
21) Proof by contradiction
22) Mathematical induction

23) Strong induction
24) Recursive mathematical definitions
25) Well orderings
26) Basics of Counting
27) Counting arguments
28) Sum and product rule
29) Inclusion-exclusion principle
30) Arithmetic and geometric progressions
31) Fibonacci numbers The pigeonhole principle
32) Permutations and combinations
33) Basic definitions
34) Pascal's identity
35) The binomial theorem
36) Solving recurrence relations
37) Common examples
38) The Master theorem
39) Graphs and Trees
40) Undirected graphs
41) Directed graphs
42) Spanning trees/forests
43) Traversal strategies
44) Discrete Probability
45) Finite probability space, probability measure, events
46) Conditional probability, independence, Bayes' theorem
47) Integer random variables, expectation
48) Law of large numbers

Course Objectives

Students will be able to:

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.

Method of Evaluation

A grading system will be established by the instructor and implemented uniformly. Grades will be based on demonstrated proficiency in subject matter determined by multiple measurements for evaluation, one of which must be essay exams, skills demonstration or, where appropriate, the symbol system.

1) Quizzes and exams that measure students' ability to use programming terminology and explain technical concepts related to computer organization.
2) Practical exams requiring students to trace programming code to predict outcome. For example, a 10-20 line section of code is provided and students must be able to identify how all of the values in all variables are altered by that code and predict the final output as specified in the code.
3) Projects requiring students to write complete application programs demonstrating knowledge of high level programming languages like: C++, Java, or Python.

Special Materials Required of Student
Flash drive

Minimum Instructional Facilities
Computer lab with networked computers and software to create a high level programming environment using C++, Java, Python, or other high level programming language.

Method of Instruction
1) Lecture and demonstration
2) Hands-on practice
3) Lab problems

Out-of-Class Assignments
1) Design, code and debug multiple programs written in one of the following languages: C++, Java, or Python.
2) Analyze instructor-assigned pre-coded programs written in one of the following languages: C++, Java, or Python, post analysis comments on the class discussion board.

Texts and References
1) Required (representative example): Discrete Mathematics and its Applications (8th ed.). Rosen. 2019.
2) Supplemental: None

Student Learning Outcomes
Upon successful completion of this course, students will be able to:
1) Utilize formal tools of symbolic logic to develop computer programs related to solving real-world problems.
2) Develop a recursion algorithm to solve a programming problem.
3) Develop solutions to problems using recurrence equations.
4) Describe different traversal methods for data structures like trees and graphs.
5) Demonstrate the differences between the binomial theorem and Bayes' theorem in a program.