Discipline: Computer Science
Originator: Mark Lehr

Riverside Community College District Integrated Course Outline of Record

Computer Science 7

CSC-7 : Discrete Structures	**College:** RIV NOR Lecture Hours: 54.000 Lab Hours: 18.000 TBA Option: Yes Units: 3.00 Letter Grade

Course Description
Prerequisite: CSC-5 or CIS-5
Course Credit Recommendation: Degree Credit

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 Set; Basic Logic; Proof Techniques; Basics of Counting; Graphs and Trees; and Discrete Probability. 54 hours lecture and 18 hours laboratory. (TBA option)
Short Description for Class Schedule
Introduction to discrete structures for Computer Science Majors. (Same as CIS-7)
Entrance Skills:
Before entering the course, students should be able to demonstrate the following skills:

1. **Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today.**
    - **CSC-5 -** Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today.
    - **CIS-5 -** Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today.
2. **Discuss the importance of algorithms in the problem-solving process. Identify the necessary properties of good algorithms.**
    - **CSC-5 -** Explain what an algorithm is and its importance in computer programming.
    - **CSC-5 -** Use pseudocode, flowcharts, and a programming language to implement, test, and debug algorithms for solving problems. Identify the information requirements, synthesize the algorithmic steps needed to transform the data input into the required output information, and organize the output format to facilitate user communication.
    - **CIS-5 -** Explain what an algorithm is and its importance in computer programming.
    - **CIS-5 -** Use pseudocode, flowcharts, and a programming language to implement, test, and debug algorithms for solving problems. Identify the information requirements, synthesize the algorithmic steps needed to transform the data input into the required output information, and organize the output format to facilitate user communication.
3. **Identify the information input requirements, synthesize the algorithmic steps needed to transform the data input into the required output information, and organize the output format to facilitate user communication.**

   - **CSC-5 -** Describe the principles of structured programming and be able to design, implement and test structured programs.
   - **CSC-5 -** Use pseudocode, flowcharts, and a programming language to implement, test, and debug algorithms for solving problems. Identify the information requirements, synthesize the algorithmic steps needed to transform the data input into the required output information, and organize the output format to facilitate user communication.
   - **CIS-5 -** Use pseudocode, flowcharts, and a programming language to implement, test, and debug algorithms for solving problems. Identify the information requirements, synthesize the algorithmic steps needed to transform the data input into the required output information, and organize the output format to facilitate user communication.
   - **CIS-5 -** Describe the principles of structured programming and be able to design, implement and test structured programs.
4. **Create computer programs in C++ using the principles of structured programming. Analyze and explain the behavior of simple programs involving the fundamental programming constructs. Modify and expand programs that use standard conditional and iterative control structures and functions. Design, implement, test, and debug programs that use fundamental programming constructs. Choose appropriate conditional constructs. Apply techniques of structured decomposition.**
   - **CSC-5 -** Apply the principles of logical and programming concepts to develop solutions for gaming, business, scientific and mathematical problems.
   - **CSC-5 -** Create computer programs using the principles of structured programming and demonstrate the use of an IDE with appropriate libraries. Design, implement, test, and debug programs that use fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and functions.
   - **CSC-5 -** Demonstrate different forms of binding, visibility, scoping, and lifetime management.
   - **CIS-5 -** Apply the principles of logical and programming concepts to develop solutions for gaming, business, scientific and mathematical problems.
   - **CIS-5 -** Create computer programs using the principles of structured programming and demonstrate the use of an IDE with appropriate libraries. Design, implement, test, and debug programs that use fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and functions.
   - **CIS-5 -** Demonstrate different forms of binding, visibility, scoping, and lifetime management.

Student Learning Outcomes:
Upon successful completion of the course, students should be able to demonstrate the following skills:

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.**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
2. **Relate the ideas of mathematical induction to recursion and recursively defined structures.**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills

about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
    - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.
3. **Analyze a problem to create relevant recurrence equations.**
    - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
4. **Demonstrate different traversal methods of trees and graphs.**
    - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
    - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.
5. **Apply the binomial theorem to independent events and Bayes' theorem to dependent events.**
    - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.

Course Content:

I. Functions, Relations and Sets
    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

II. Basic Logic
    1. Propositional logic
    2. Logical connectives
    3. Truth tables
    4. Normal forms (conjunctive and disjunctive)
    5. Validity
    6. Predicate logic
    7. Universal and existential quantification
    8. Modus ponens and modus tollens
    9. Limitations of predicate logic

III. Proof Techniques
   1. Notions of implication, converse, inverse, contrapositive, negation, and contradiction
   2. The structure of mathematical proofs
   3. Direct proofs
   4. Proof by counterexample
   5. Proof by contradiction
   6. Mathematical induction
   7. Strong induction
   8. Recursive mathematical definitions
   9. Well orderings

IV. Basics of Counting
   1. Counting arguments
   2. Sum and product rule
   3. Inclusion-exclusion principle
   4. Arithmetic and geometric progressions
   5. Fibonacci numbers
   6. The pigeonhole principle
   7. Permutations and combinations
   8. Basic definitions
   9. Pascal's identity
   10. The binomial theorem
   11. Solving recurrence relations
   12. Common examples
   13. The Master theorem

V. Graphs and Trees
   1. Trees
   2. Undirected graphs
   3. Directed graphs
   4. Spanning trees/forests
   5. Traversal strategies

VI. 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

Additional Laboratory Content

Lab activities for this course support the lecture through the use of practice activities, software simulations, and streaming video under the supervision and guidance of faculty.

1. Basics of Counting Lab:
   Develop programs that simulate the following mathematical principles:
   1. Sum and product rule
   2. Arithmetic and geometric progressions
   3. Fibonacci numbers
   4. The pigeonhole principle
   5. Permutations and combinations
   6. Pascal's identity
   7. The binomial theorem
   8. Solving recurrence relations
2. Graphs and Trees Lab:
   Create programs that traverse Graphs

        1. Traversal strategies
3. Discrete Probability Lab:
   Simulate distributions and show how they correspond with theoretical outcomes:
    1. Finite probability space, probability measure, events
    2. Conditional probability, independence, Bayes' theorem
    3. Integer random variables, expectation
    4. Law of large numbers

Methods of Instruction:
Methods of instruction used to achieve student learning outcomes may include, but are not limited to, the following activities:

- Class lectures, discussions, and demonstrations of formal logic, proofs, recursion, analysis of algorithms, sets, combinatorics, probability theory, number theory, relations, functions, matrices, graphs, trees, and Boolean algebra.
- Drills and pattern practices utilizing hand-outs and/or computer-based tools in order to assist the students in mastering the techniques involved in applying the discrete mathematical principles and techniques to the solution of applications related the topics mentioned above.
- Provision and employment of a variety of learning resources such as videos, slides, audio tapes, computer-based tools, manipulatives, and worksheets in order to address multiple learning styles and to reinforce material.
- Pair and small group activities, discussions, and exercises in order to promote mathematics discovery and enhance problem solving skills.

Methods of Evaluation:
Students will be evaluated for progress in and/or mastery of student learning outcomes using methods of evaluation which may include, but are not limited to, the following activities:

- Computer programs designed to demonstrate the acquisition of an understanding of formal logic, proofs, recursion, analysis of algorithms, sets, combinatorics, probability theory, number theory, relations, functions, matrices, graphs, trees, and Boolean algebra.
- Quizzes/examinations designed to measure students' degree of mastery of the mathematical theories that forms the foundation of computer science.
- Collaborative projects designed to demonstrate successful understanding and application of the mathematical theories that forms the foundation of computer science.
- Computer Laboratory assignments/projects designed to clarify students' individual understanding of formal logic, proofs, recursion, analysis of algorithms, sets, combinatorics, probability theory, number theory, relations, functions, matrices, graphs, trees, and Boolean algebra strengths and areas of improvement related to these skills.
- Final examination designed to evaluate students' overall achievement of course objectives discrete mathematics.

Sample Assignments:
Outside-of-Class Reading Assignments

- Read and understand college level texts concerning recursion, algorithms, combinatorics, number theory, matrices, graphs and trees as applies to Computer Science.

Outside-of-Class Writing Assignments

- There will be several assignments where the student demonstrates with computer code a thorough understanding of the formal logic of computer science. These algorithms/code will cover proofs, recursion, combinatorics, relations, functions, graphs, trees, Boolean algebra and modeling.

Other Outside-of-Class Assignments

- The outside of class assignments will be the completions of algorithms/code development started in class after the topics are presented.

Course Materials:
All materials used in this course will be periodically reviewed to ensure that they are appropriate for college level instruction. Possible texts include the following:

- Epp, S.. *Discrete Mathematics with Application's.* 4th Brooks-Cole, 2010.
- Gersting, J.. *Mathematical Structures for Computer Science.* 7th Freeman, 2014.
- Lipschutz, S., Lipson, M.. *Schaum's Outline Discrete Mathematics.* 3rd Schaum, 2009.

Codes/Dates:
CB05 NOR Transfer Status: Transfers to Both UC/CSU (A)
CB05 RIV Transfer Status: Transfers to Both UC/CSU (A)
C-ID#: COMP 152
Board of Trustees Approval Date: 01/22/2013
COR Rev Date: 01/22/2013
Generated on: 2/12/2018 10:12:18 AM