Computer Science 261
Discrete Structures
Fall 2020

Instructor:  Dr. Chris Healy
My office is located in Room 200-I in Riley Hall.  My office hours are MWF 9:00 – 11:00,
R 2:00 – 2:50 and also by appointment.  Please see me if you ever have any questions
during the course.  I can be reached by phone (294-2233) and e-mail
(chris.healy@furman.edu).

Class meetings:  MWF 8:00 – 8:50 a.m. and Thursday 3:00 – 5:00 p.m., 106 Riley Hall

Purpose
Many problems that we need to solve in computer science have a significant quantitative or
logical component.  The purpose of this course is to acquaint students with quantitative
techniques that are used extensively in computer science.

The prerequisite for this course is CS 121 (Introduction to Computer Programming).  This
class leads into several other courses in the department:  CS 361 (Data Structures), CS 343
(Artificial Intelligence), CS 344 (Computer Graphics) and CS 461 (Computational Theory),
as well as Math 330 (Combinatorics and Graph Theory).

Web site:  Notes and handouts can be found here:  http://cs.furman.edu/~chealy/cs261

Textbook:  *Discrete Mathematics with Applications*, by Susanna Epp, Brooks/Cole
Publishers, fifth edition, 2019.

Grade calculation
    25%  Homework
    25%  Test #1 – Thursday, September 17, 3:00 – 5:00 p.m.
    25%  Test #2 – Thursday, October 22, 3:00 – 5:00 p.m.
    25%  Final exam – Wednesday, December 2, 8:30 – 11:00 a.m.

Please note the dates/times of these exams.  Any appropriate documentation supporting
special arrangements necessary for any test must be given to me within the first week of
class.  If you are authorized to take an exam remotely, then you are required to send your
exam paper to me electronically by the end of the class period.

Homework
You should expect a short homework assignment approximately twice a week.  An
assignment will usually consist of some questions from the textbook and will be due at the
beginning of class on the next class day.  You may not turn in homework late since we will
be going over the answers at the beginning of class.  These written homework exercises will
be graded primarily on the basis of thoroughness, proper technique, clarity, and neatness.

When you work homework problems, you should begin by restating the problem in your
own words.  This will help you think about ways to solve the problem and how it may
compare to others you may have seen.  It is also a check that you are interpreting the
problem correctly.  In homework solutions, use complete sentences throughout.  The final
answer is not as important as how you arrived at it and how you communicate your
approach and the steps you take.

Please note the following homework policy, because it may differ from other classes you have had.  Each student is required to turn in an individual homework submission.  Getting help from someone else or collaborating with another student when working on homework is allowed.  If you do get help, please include a statement at the end of your homework paper that says who you worked with, which questions you needed help with, and to what extent you still do not understand how to do the problem(s).  Your homework grade will not be affected by this statement.  Examples of such a statement might be:  "Donald Duck and I worked on the second problem together, and now I fully understand how to do it."  Or, "I did not know how to do the last problem.  Mickey Mouse showed me how to do it, but I still would not be able to do it myself."

If you don't include a statement of assistance at the end of your homework, I will assume that you were able to complete all of the problems by yourself.  The reason why I am tracking whether you needed help on certain problems is so that I can contact you about what you don't understand.  My mission is to help you master all of the material of the course.

Plagiarism means receiving assistance without the proper attribution, and this will not be allowed.  Also, if you are using my collaboration policy merely as a way for other people to do your homework for you, this will also not be allowed.  Cheating penalties will be substantial, up to and including an automatic failing grade in the course.  The minimum penalty for cheating related to homework will be a failing grade on the assignment.

Attendance
If you are attending the class remotely, it is your responsibility to make your presence known to the instructor.  If you are absent from a test, you will earn a score of zero, unless your absence is excused.  Excused absences include illness (doctor's note elaborating your incapacitation will be required), circumstances beyond your control, or a required extra-curricular activity in which you are an official representative of the university.  In any case, you are required to submit written documentation to excuse an absence.  If your absence is excused, then I will assign you to a make-up exam.

Preparation
I estimate that you will need to study up to 5 hours per week for this class.  Studying on a consistent daily schedule is more important than the number of hours.  Study includes reviewing and organizing your notes, working problems from the textbook, becoming acquainted with the material to be discussed in the next class, and completing homework assignments.  Don't forget the most important thing – I am here for you.  Please come to my office anytime for help or advice in this course.

Course Objectives
Upon successful completion of this course, students should be able to do the following.

Logic

1. Understand the semantics of the logical operators AND, OR, NOT and implications.  Write truth tables for logic/Boolean expressions.  Determine if a statement form is a tautology or contradiction.  Draw and interpret digital circuits that use logic gates.  Demonstrate that NAND and NOR are universal gates.  Be able to convert any Boolean expression into one that uses only NAND or only NOR operations.
2. Simplify or rewrite Boolean expressions using DeMorgan's law and other algebraic properties.

3. For an implication, be able to write its negation, converse, inverse and contrapositive. Recognize that the contrapositive is equivalent to the original.
4. Given a logical argument in propositional logic (in either logic symbols or English), determine whether the argument is valid or invalid.
5. Design combinational circuits that may have several inputs and several outputs. Write a Boolean function as a sum of minterms. Simplify Boolean functions using a Karnaugh map or the Quine-McCluskey technique.
6. Given a singly or multiply quantified statement, be able to convert between English and logic symbols.
7. Determine whether a quantified statement is true or false, and be able to argue which is the case, including finding counterexamples. Be able to negate propositional and quantified statements.

Proof Methods and Analysis of Algorithms

8. Exhibit familiarity with the technique of direct proof, for example showing some given elementary definition or property is satisfied. Apply other proof techniques such as proof by contradiction and proof by contraposition.
9. Prove assertions using the principle of mathematical induction, and be able to apply this technique specifically to verify statements about summation formulas, inequalities, divisibility, proving the correctness of a loop, and some miscellaneous cases to the extent it is clear the student is doing more than just memorizing isolated techniques, for the purpose of being able to apply mathematical induction in an unfamiliar yet still simple context.
10. Understand floor and ceiling functions. Use the definition of floor and ceiling to convert between a floor or ceiling expression and an inequality.
11. Evaluate multiple (nested) sigma notation. Given a nested loop, calculate the number of operations performed using Bernoulli formulas.
12. State and apply the definition of big-O notation. Also be able to recognize and show that $f(n)$ is not $O(g(n))$ in some cases. Discuss why a given algorithm may have a logarithmic order of complexity such as $O(n \log n)$. Identify what specific effect that changing the input size will have on the performance of the algorithm.

Sets

13. Determine the union, intersection, complement and Cartesian product of given sets. Verify an alleged property of sets, including subsets.
14. Use Venn diagrams or Karnaugh maps to determine the number of elements in a set or subset. Apply the principle of inclusion and exclusion.
15. Perform operations on sets of strings, including concatenation and Kleene star.
16. Recognize that set theory and propositional logic are Boolean algebras. Apply DeMorgan's law and other properties of Boolean algebras.
17. Use bit vectors and bitwise logical and shift operators to implement set/logical operations. Use bitwise operators in a useful application such as steganography.

Counting, Probability and Generating Functions

18. Given a situation describing a permutation or combination, be able to distinguish which is the case. Be able to solve permutation and combination problems, such as: the fundamental counting principle, factorials, distinguishable permutations, indistinguishable subsets, balls in urns, breaking a problem into cases. Understand and apply the recursive nature of Pascal's triangle and Stirling numbers. Define probability as a fraction, and compute probabilities based on permutation or combination situations.

19. Use generating functions to solve combination problems.

## Recursion and Recurrence Relations

20. Evaluate recursive formulas, and write recursive definitions.  For a given set of numbers or strings defined recursively, use induction to show that some property is satisfied.  Analyze an algorithm by writing a recursive formula (recurrence relation).
21. Solve recurrence relations using the method of undermined coefficients.  Solve non-homogeneous recurrence relations.
22. Use induction to show that a given explicit formula is equivalent to its corresponding recurrence relation.

## Relations and Functions

23. Depict a relation as an adjacency matrix.  Given a relation, determine if it is:  reflexive, symmetric, antisymmetric, transitive – and be able to appeal to the definitions to explain why this is the case.  Conversely, give an example of a relation that satisfies these types of criteria.  Determine if a relation is a partial order or equivalence relation.  For an equivalence relation, identify the equivalence classes.  Explain equivalence relations in terms of partitions of a set.
24. Determine if a relation is a function.  Determine the domain, co-domain and range of a function.  Determine if a function is one-to-one and/or onto.
25. Design and analyze simple finite automata.
26. Show whether an infinite set is countable or uncountable.
27. Design applications of functions such as cryptography and error-correcting codes.

## Graphs

28. Draw a graph that has a given degree sequence, or explain why such a graph does not exist.
29. Determine if two graphs are isomorphic.  Determine if a graph is bipartite.
30. Identify paths in a graph.  Find Euler and Hamiltonian cycles in a graph.
31. Show how a graph can be represented as an adjacency matrix.  Explain the significance of the values in the matrix, particularly values other than 0 or 1.
32. Discuss applications of graph theory at a high level, explaining why it is helpful to conceptualize certain entities in terms of nodes and edges.  For example:  as a data structure, use in compiler design to detect loops, networking, finite automata, etc.
33. Construct a Huffman code given frequency data on what is to be encoded.  Also be able to encode and decode the data.  Calculate the length of a message that is encoded this way.  Give an example of an ambiguous code.
34. Perform preorder, inorder and postorder traversals of a binary tree.  Be able to represent a mathematical expression as a tree and write prefix, infix and postfix notation.  Evaluate prefix and postfix expressions.
35. Perform Dijkstra's shortest path algorithm on a weighted graph.
36. Create spanning trees using depth-first search and breadth-first search.  Apply Kruskal's or Prim's algorithm to find a minimal spanning tree.  Distinguish between the minimum spanning tree problem and the traveling salesman problem.

***Tentative course schedule

| Week | Days | Book sections | Topics |
|---|---|---|---|
| 1 | – – 19 20 21 | 2.1 – 2.3 | Logic, statements, operators, truth tables, DeMorgan's law, tautology, contradiction, implication, alternate forms, negating, necessary & sufficient conditions; arguments |
| 2 | 24 – 26 27 28 | 2.4 | Liar puzzle; Logic gates, boolean functions, NAND & NOR universal gates<br>Digital design procedure, don't cares, Karnaugh maps |
| 3 | 31 – 2 3 4 | 3.1 – 3.3, 4.1 | Quine-McCluskey; begin quantified statements<br>Quantified statements, predicates, negating, translating/English<br>Two quantifiers, practice<br>Start proof unit: direct proof, odd/even, rational, divides |
| 4 | 7 – 9 10 11 | 4.3, 4.5, 4.6, 5.1, 5.2 | Floor, ceiling; indirect proof (contradiction/contraposition)<br>Series, Bernoulli formulas, counting loop iterations<br>Induction for summations, linear combinations (coin/stamp) |
| 5 | 14 – 16 17 18 | 5.3 5.5 | Induction for divisibility, inequalities<br>Test #1 on Thursday<br>Loop correctness |
| 6 | 21 – 23 24 25 | 6.1, 6.2 9.1 – 9.2 | Start unit on sets: defining a set, operations, empty set<br>Sets of strings<br>Boolean algebras; bit vectors and bitwise operations<br>Counting unit: multiplication rule, distinguishable permutations |
| 7 | 28 – 30 1 2 | 9.3, 9.5 – 9.7 | Inclusion/exclusion, breaking into cases, permutations with restrictions; combinations<br>Card examples, indistinguishable subsets, ball-in-urn<br>Pascal's triangle, Stirling numbers |
| 8 | 5 – 7 8 9 | 9.8 | Counting passwords, expected value<br>Practice counting and probability problems<br>Generating functions |
| 9 | 12 – 14 15 16 | 7.1, 12.1, 12.2, 7.2, 7.4 | Function terminology, formal definition, examples in CS<br>Finite versus infinite, countable versus uncountable<br>Formal language, finite automata<br>One-to-one, onto |
| 10 | 19 – 21 22 23 | 5.6, 5.7 | Start unit on recursion and recurrence relations: recursive functions<br>Test #2 on Thursday<br>Basic iteration method to solve recurrences |
| 11 | 26 – 28 29 30 | 5.8 | Homogeneous 2nd order recurrence relations<br>Proving explicit formula = recursive formula<br>Inhomogeneous recurrence relations |
| 12 | 2 – 4 5 6 | 8.1 – 8.3, 8.5 | Relations unit: reflexive, symmetric transitive, other properties, modulo |
| 13 | 9 – 11 12 13 | 1.4, 10.1 – 10.4 | Start graphs unit: definition, applications, adjacency matrix, degree sequence<br>Bipartite, isomorphism<br>Intro to Trees |
| 14 | 16 – 18 19 20 | 10.5 | Huffman code, binary search tree<br>Breadth first search, depth first search, expression as tree<br>Tree traversals, weighted graphs<br>Directed graphs |
| X | – – 2 – – | – | Final Exam 8:30 Wednesday |

Guiding principles for students in Computer Science classes

1.  We are here to learn and explore.
    a.  Seek discussions with the instructor and classmates about the material to reinforce your understanding, practice communicating ideas.
    b.  Have fun.  Live in the moment (i.e. don't dwell too much on the difficulties of yesterday or tomorrow).  Enjoy the journey and intellectual feast.  Be enthusiastic about what you are doing.

2.  You <u>can</u> be successful in this class.  Every day is an opportunity for an epiphany.  Don't let mistakes or setbacks hold you back.  After some effort, things can suddenly click in your mind.  Your eagerness to learn is more important than intellectual ability in determining your success in this course.

3.  Learn by doing, not just passively reading, listening or watching.
    Each study period needs to have a clear goal.
    Pay attention to the big picture and the facts that you are collecting.

4.  Be organized.  Take notes on what you read.  Review earlier material as needed.  Create a cumulative study outline every few weeks.  Maintain a portfolio of your work.

5.  Be patient when solving a homework or lab problem.
    a.  There is no need to rush.
        Don't worry if your first attempt at a solution is wrong.
        Read all instructions and be methodical.
        Take time to gather your thoughts.
        Deliberately write out your thought process and plan of attack.
    b.  A computer program or other homework assignment may take up to several hours to complete.  In programs you need to comment your code as you go, because you will quickly forget what looks obvious right now!  Realize that you don't need to finish everything in one sitting.
    c.  Break up large problems into small, more manageable pieces.
    d.  Don't get bogged down with too many mechanical details.  Computing is all about removing tedium from routine tasks.

6.  Be curious, and always ask questions.
    a.  Find a topic or application that you are enthusiastic about.
    b.  Consider alternative solutions to a problem.
    c.  When finishing a problem, ask yourself if this problem or its solution lends itself to other problems.

7.  Computer science is about logic, structured thinking, information, communication and problem solving.  Thus, it has connections to many other fields in the sciences, humanities and social sciences.  You will find the analytical techniques useful in your career.