CSCI 243

Discrete Mathematics and Functional Programming

Fall 2011 MFW 2:00-3:05 pm SCI 131

http://csnew.wheaton.edu/~tvandrun/cs243

Thomas VanDrunen

☎630-752-5692 630-639-2255 ✉Thomas.VanDrunen@wheaton.edu

Office: SCI 163 Office hours: MThF 3:15-5:00 pm; Th 9:15-11:00 am.

Contents

CATALOG DESCRIPTION. Sets, logic, the nature of proof, induction, algorithms, algorithm correctness, relations, lattices, functions, and graphs. Functional programming and recursion using the ML programming language.

OBJECTIVES. The chief goal of this course is to teach you formal reasoning, practiced under two heads: mathematical proofs and computer programs. At the end of this course you should be able to

- Manipulate symbolic logical forms.
- Write mathematical proofs, especially for results from basic set theory.
- Write simple programs in the ML programming language.

Other themes include

Writing and using formal definitions. We look carefully at how to define formal, rigorous definitions of mathematical ideas, built from primitive terms.

Thinking recursively. *Recursion* is defining something in terms of itself. This technique is crucial both to programming and to some kinds of mathematical definitions and proofs.

Analysis and synthesis. Many of our proofs and programs comprise two main steps: breaking something apart and putting something else together.

OUTLINE. This course is organized under the following headings:

Set and List. We meet the basic mathematical concepts of set, element, set operations, cardinality, Cartesian products, and powersets. We begin the basics of the ML programming languages including functions and datatypes. We learn to use ML's main composite type, the list.

Proposition. We explore the system of boolean logic (the "first-order predicate calculus"). This heading is characterized by three "games" to exercise your understanding of symbolic logic: 1. verifying logical equivalences; 2. verifying argument forms; 3. verifying argument forms with quantification. We also write ML programs that use boolean operators and consider how the quantification of a program specification affects the algorithm to solve it.

Proof. This is the turning point of the semester, perhaps the most important heading. We learn to write careful mathematical proofs of set-theoretical propositions. This includes one of the most challenging sections, proofs about powersets. We also consider the connections between proofs and algorithms.

Relation. We build on our understanding of sets to consider the definition of mathematical relations and their properties, propositions about them, and programs that manipulate them. Relations are useful concepts in themselves, but this heading also gives us opportunity to practice further the proving and programming techniques from earlier in the course.

Self reference. Earlier parts of our study will have introduced recursive definitions, but here we take the idea head-on. Specific topics are recursive types in ML programming and proofs using structural and mathematical induction.

Function. We study functions as mathematical objects built on set theory, as we will have done for relations. The proofs in this section are an apex of the mathematical topic stream. We also learn idioms in ML programming based on the theory of functions.

Your choice. At the end of the semester, you will choose one of several topics for a self-directed study.

For a detailed outline, see the table of contents in your textbook. For a schedule, see the course website.

Course procedures

How we do this course. This course has a pretty predictable rhythm to it. Class time is mainly for working out new kinds of problems together. There will be daily assignments. Tests come when we get to good stopping points.

Before class you are to read the assigned sections from the textbook. In class I will review and highlight material, especially definitions, that you have read. I will avoid lecturing, preferring to devote most of our meeting time to practicing sample problems. Some class periods will also include demonstrations of new ML features. At the end of each class I will assign problems from the book, which will also be posted on the course website. These will be a mix of pencil-and-paper exercises for the math portion (mostly proofs) and programming exercises for the computer science portion (the latter to be turned in by email).

Readings. It is important that you read the assigned sections for each day. The readings fit into three categories.

Read carefully means that I will not cover that material in class at all. It's background stuff for what we really want to talk about. You're solely responsible for it.

Read means that I will highlight and review the main ideas but not lecture on them. I will assume you have seen them before. We will work on sample problems from those sections in class.

Skim means that this is very difficult material that most students will need to see twice to understand. Familiarize yourself with it first, and then I will lecture on it in class.

Project. For the last two weeks of the course you will choose from one of the topics below (these are extra chapters in the book that will be available online) and work through it. You may work together with one or two other people. Class time will be used to meet with me on your progress.

Available chapters: Graph, Complexity Class, Lattice, Group, Automaton.

Grading. There will be three tests (scheduled for Sept 23, Oct 21, and Nov 18, subject to change) and a final (Thursday, Dec 15 at 8:00 am).

instrument	*weight*
Homework	22
Test 1	13
Test 2	13
Test 3	13
Project	10
Final exam	29

I will also give one point of extra credit (applied towards homework) for every mistake you find in the textbook, if you are the first to discover it. Bigger suggestions about the presentation (like new exercises and examples or ways to make a section more understandable) will be rewarded appropriately.

Follow this course on Twitter. You are encouraged to get a Twitter account and follow `@TVD_CSCI243` for announcements like changes to the schedule. If you don't use Twitter and strongly do not want to start, email me and I'll add your email address to an alternate list for announcements.

How to succeed in this course. By this point in your academic career you should have developed good study habits and found what works best for you. In my experience, however, its seems many students could still use a few pointers.

Prepare for class. Set aside time the day before or in the morning to think about what we will be covering. Take the readings seriously. Try some of the exercises in the sections before we cover them in class.

Take the right amount of notes. You need to be active in class, working through the problems we're doing on the board. That said, some of you need to go easy on the note-taking. I feel sorry for the students who seem to think that their main task in class is to transcribe everything written on the board. So busy writing, they don't have time to process what's going on in class. I wrote the course manual in a way that should minimize the need to take notes. I'd rather you put your energy into *thinking*.

Keep up with the material. The material in this class keeps on building on itself. If you don't understand something, don't just shrug it off and move on. Even if it doesn't seem like last week's material is being used this week, last week's material is going to come back later.

When all else fails, *ask for help.* A lot of learning in a class like this happens during office hours.

Policies etc

Academic Integrity. Students are encouraged to discuss homework problems and ideas for solutions. However, your solutions, proofs, and programs must be your own. If you are having trouble debugging a program you have written, you may show it to a classmate to receive help; likewise you may inspect a classmate's incorrect program to give help. However, you should not show *correct* code to a classmate, nor should you look at another student's correct code, to give or receive help. Homework on which students have unfairly collaborated will not be accepted.

Assignments. Unless otherwise specified, assignments are due at the class period after it was assigned. I will collect the assignments at the end of class. However, you are granted an automatic grace period until 5:00 pm that day. Assignments not complete by class time can be put in the instructor's box. If you have not completed the assignment by the end of the grace period (5:00 pm), then turn in what you have at that time for partial credit.

Attendance. Students are expected to attend all class periods. It is courtesy to inform the instructor when a class must be missed.

Examinations. The final exam is Thursday, Dec 15, at 8:00 AM. I do not allow students to take finals early (which is also the college's policy), so make appropriate travel arrangements.

Special needs. Whenever possible, classroom activities and testing procedures will be adjusted to respond to requests for accommodation by students with disabilities who have documented their situation with the registrar and who have arranged to have the documentation forwarded to the course instructor. Computer Science students who need special adjustments made to computer hardware or software in order to facilitate their participation must also document their needs with the registrar in advance before any accommodation will be attempted.

Office hours. I try to keep a balance: Stop by anytime, but prefer my scheduled office hours. I consider 3:15–5:00 pm everyday to be my unofficial office hours this semester, but on some Tuesdays and Wednesdays I will have conflicts during that time. Also, any time my door is

closed, it means I'm doing something uninterruptable, such as making an important phone call. Rather than knocking, please come back in a few minutes or send me an email.

Dress and deportment. Please dress in a way that shows you take class seriously—more like a job than a slumber party. (If you need to wear athletic clothes because of activities before or after class, that's ok, but try to make yourself as profession-looking as possible.) If you must eat during class (for schedule or health reasons), please let the instructor know ahead of time; we will talk about how to minimize the distraction.

Electronic devices. Please talk to me before using a laptop or other electronic device for note-taking. I will discourage you from doing so; if you can convince me that it truly aides your comprehension, then I will give you a stern warning against doing anything else besides note-taking. Trying out programming concepts on your own during class time is not productive because it takes you away from class discussion. You cannot multi-task as well as you think you can. Moreover, please make sure other electronic devices are silenced and put away. ***Text in class and DIE.***