CSCI 243

Discrete Mathematics and Functional Programming

Fall 2022   MFW 2:15-3:25 pm   MEY 131

`http://cs.wheaton.edu/~tvandrun/cs243`

Thomas VanDrunen

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

Office: MEY 163   Office hours: Drop-in: MWF 3:30–4:30pm; Or by appointment through Calendly

---

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.

TEXTBOOK. Thomas VanDrunen, *Discrete Mathematics and Functional Programming*, Franklin, Beedle and Associates, 2013.

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.

This course also fulfills the AAQR thematic core tag in the Christ at the Core curriculum. In terms of the AAQR learning outcomes at the end of this course you should be able to

- Capture and model phenomena in nature, culture, and the human-built world using sets, relations, and functions as well as ML datatypes. (AAQR LO # 1)

- Devise, implement (in the ML programming language), and test solutions to algorithmic problems, using symbolic logic (especially quantification) to analyze the problem and synthesize a solution. (AAQR LO #2)

- To write formal proofs for propositions about sets, relations, functions, and the correctness of algorithms. (AAQR LO #3)

Finally, together we have the general objective of seeing proof-writing and program-writing as a creative expression for God's glory and observing mathematical logic as part of God's creation.

THEMES. Three themes pervade the various topics of this course.

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.

Other. At the end of the semester, we will briefly cover a few extra (but important) programming topics.

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. Before class you will read sections from the book on the day's topic and take a quiz on Canvas to enforce the reading. In class we will highlight the main ideas from the reading and work through sample problems together. After class you will complete the assigned problem sets for the next class. Occasionally there will be supplemental videos to reinforce specific concepts.

ELECTRONIC COURSE ORGANIZATION. Course material (assignments, quizzes, slides, videos, etc) can be found on Canvas; additionally, the course organization and some of the material can be seen through a course website I have made which presents the course as in a calendar format. Assignments are to be submitted through Canvas.

I use the "Gradebook" feature on Canvas only to communicate scores on individual assignments and tests. I do **not** use the Canvas gradebook for my official record-keeping for scores, for calculating semester scores, or determining letter grades. Please **ignore** any grade estimate that Canvas gives you for this course.

READINGS. It is important that you read the assigned sections for each day.

Some of the readings you'll be told to *read carefully.* This means I won't lecture on it. Instead, after reviewing the main points, we'll work through problems and examples together.

Other readings you'll be told to *skim*. This means I will lecture on it, but you'll understand it better if you come to class with a general idea of where we are heading. Note well that *skim* does not mean *skip*.

ASSIGNMENTS. For (almost) every class session, there will be an assignment to be done for the next session. Programming problems (unless otherwise noted) should be turned in through a special page for submitting programming problems; a link to this page can be found on Canvas and the course website. All other problems are to be turned in through Canvas.

QUIZZES. A typical quiz will be two questions long, take no more than five minutes, and assess whether the student has done the reading. Quizzes will be posted to Canvas. I encourage you to take the quizzes without looking at notes or the book, but since that isn't practical to enforce, it isn't an absolute rule.

TESTS. There will be three tests (scheduled for Sept 25, Oct 27, and Nov 29, subject to change) and a final (Tues, Dec 12, 1:30 pm–3:30 pm).

PROJECT. This course has a term project in which you apply the discrete math material and what you have learned about programming to model real-world information or phenomena using ML. Work on the project will be spread throughout the semester. Details of the assignment can be found on the course website.

SUBMISSION OF ASSIGNMENTS. As noted above, programming problems in assignments are to be submitted in through a special turn-in page. All other work should be turned in on paper. I will collect homework at the end of each class session; you may also turn homework into the mesh box by my office door.

GRADING. In order to pass the course (that is, to receive a D grade or better), a student must (a) turn in a complete project, (a) achieve an average score of at least 50% on the tests and final, **and** (b) achieve at minimum score of 40% on the final.

For students who have met the minimum requirements, their semester score will be calculated as the weighted average of the following individual scores (each as a percentage):

instrument	weight
Homework and quizzes	20
Project	10
Test 1	15
Test 2	15
Test 3	15
Final exam	25

Letter grades (A–D) will be determined by clustering of students' semester scores.

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; they make themselves so busy writing, they don't have time to process what's going on in class. I wrote the textbook in a way that should minimize (but not eliminate) 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—get it right. You can use the re-turn-in policy for assignments (see below) to get credit for this. 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 things aren't working, *ask for help.* A lot of learning in a class like this happens during office hours. Specifically, *ask for help right away.* Do not get behind in this course. The ideas we explore are tightly dependent on preceding material. It is very hard to catch up in this course if you get behind on understanding the concepts.

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.

Students should not use Copilot, ChatGPT, or similar tools for programming or written assignments. Students certainly may compare their solution to a solution produced by tools like this *after* they have received your score, but these tools should not be used on any work to be turned in.

Homework on which students have unfairly collaborated will not be accepted. Repeated offenses will be handled through the college's disciplinary procedures and may result in failing the course. (See also the College's statement below.)

ASSIGNMENTS. Unless otherwise specified, assignments are officially due at the class period after it was assigned. However, you are granted an automatic grace period until 5:00 pm that day. 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. Assignments are spot-checked: depending on the assignment, around half of the problems will be graded, and you won't know ahead of time which ones. (This is for practicality—the TAs and I don't have time to grade every problem.)

Unless otherwise specified, coding problems are to be turned in using the automated grader at `http://cs.wheaton.edu/~tvandrun/cs243/ml-grading.php`. The same rules about due dates and turn-in times apply. The system will run the submitted code against a series of test cases and will report on the results. (You will not see the test cases themselves. Testing your code with your own test cases is part of the assignment.) You may submit as many times as you want until the deadline. Submissions that do not pass all the test cases will be assigned partial credit by the instructor or TA, reported by email.

RE-TURN-IN OF ASSIGNMENTS. After you receive your graded assignments back, you may redo any proof, programming problem, or "game" problem (from Chapter 3) that you did not receive full points on and turn it back for regrading no more than two class meetings later. The regraded problems will be evaluated by the same criteria as originally used, and the student may earn back up to full credit for those problems. The same policy applies to regraded problems when they are turned back: if the student does not receive full credit on a re-turned-in problem, he or she may try again (indefinitely). For any re-turn-in, please also include the graded original (and any subsequent graded attempts).

Some details: For paper assignments, "two class meetings" means at the end of class two class days later. For example, if an assignment is turned back on Monday, the student must re-turn-in the assignment by the end of class Friday. Students do not receive extra time to redo problems if they are delayed in receiving the graded assignment because of absence or lateness—time is measured by when you would have received it back, not when you actually did. Since programming assignments are submitted electronically and graded automatically, students have opportunity to re-turn-in a problem only if they have submitted a good-faith attempt by the original due date. The instructor or TA will send them a partial-credit assessment by email. Students then have three weekdays (72 hours) while the college is in session from the time that the TA emails an assessment of partial credit to re-turn-in a solution to the problem. For example, if a student receives a graded homework by email at 2:16 am Thursday, the student must re-turn-in the assignment no later than 2:16 am Tuesday. These are all accounted on a per-problem basis. Any problem for which the "two class days" period has elapsed is no longer eligible for re-turn-in; similarly, problems for which no attempt was turned in for the original due date are not eligible for re-turn-in (students may still turn in such problems for correction and comments, but not for credit). In any case, the opportunity applies only to answers that have an error of substance; answers with only a minor mistake that is completely corrected by the grader's comments may not be resubmitted (for proofs, this would apply to answers with only .25 point deduction).

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

EXAMINATIONS. Students are expected to take all tests, quizzes, and exams as scheduled. In the case where a test must be missed because of legitimate travel or other activities, a student should notify the instructor no later than one week ahead of time and request an alternate time

to take the test. In the case of illness or other emergency preventing a student from taking a test as scheduled, the student should notify the instructor as soon as possible, and the instructor will make a reasonable accommodation for the student. The instructor is under no obligation to give any credit to students for tests to which they fail to show up without prior arrangement or notification in non-emergency situations. The final exam is Tues, Dec 12, 1:30 pm–3:30 pm. Students are not allowed to take the final exam at a different time (except for urgent reasons, approved by the department chair, as per the college's policy), so make appropriate travel arrangements.

ACCOMMODATIONS. If you have a documented need for accommodations, I will have received a letter on your behalf from the Disability Services Office. But *please talk to me* about what accommodations are most useful to you. In particular, if you desire accommodations for test-taking, talk to me a reasonable amount time in advance (say, at least two class periods) so arrangements can be made. (See also the College's statement below.)

OFFICE HOURS. My *drop-in* office hours this semester are MWF 3:30–4:30pm. You can make an appointment through Calendly; I'm available most of the day on Thursday and sometimes on other days.

ELECTRONIC DEVICES. My intent is for class to be an electronic-device-free zone. Please keep all devices (laptop, tablet, and phone) put away and silenced. If you absolutely need to check your phone for something, please discreetly step out in to the hall. **NO TEXTING OR SOCIAL MEDIA USE IN CLASS.**

All this, the Lord willing.

College syllabus statements

<small>THE COLLEGE REQUIRES THAT THE FOLLOWING STATEMENTS BE INCLUDED IN ALL SYLLABI.</small>

The "Academic Information" website referred to below is found and `https://catalog.wheaton.edu/undergraduate/academic-policies-information/academic-information/`

ACADEMIC INTEGRITY. (See "Integrity of Scholarship" on "Academic Information" website.)
The Wheaton College Community Covenant, which all members of our academic community affirm, states that, "According to the Scriptures, followers of Jesus Christ will be people of integrity whose word can be fully trusted (Psalm 15:4; Matt. 5:33-37)." It is expected that Wheaton College students, faculty and staff understand and subscribe to the ideal of academic integrity and take full personal responsibility and accountability for their work. Wheaton College considers violations of academic integrity a serious offense against the basic meaning of an academic community and against the standards of excellence, integrity, and behavior expected of members of our academic community. Violations of academic integrity break the trust that exists among members of the learning community at Wheaton and degrade the College's educational and research mission.

ACCOMMODATIONS. (See "Learning and Accessibility Services" on the "Academic Information" website).
Wheaton College is committed to providing access and inclusion for all persons with disabilities, inside and outside the classroom. Students are encouraged to discuss with their professors if they foresee any disability-related barriers in a course. Students who need accommodations in order to fully access this course's content or any part of the learning experience should connect with Learning and Accessibility Services (LAS) as soon as possible to request accommodations `http://wheaton.edu/las` (Student Services Building -Suite209, `las@wheaton.edu`, phone 630.752.5615). The accommodations process is dynamic, interactive, and completely free and confidential. Do not hesitate to reach out or ask any questions.

BEHAVIOR POLICY. (See "Classroom Demeanor" on the "Academic Information" website).

GENDER-INCLUSIVE LANGUAGE. (See "Gender Inclusive Language" on the "Academic Information" website).
Please be aware of Wheaton College's policy on inclusive language, "For academic discourse, spoken and written, the faculty expects students to use gender inclusive language for human being."

TITLE IX AND MANDATORY REPORTING. Wheaton College instructors help create a safe learning environment on our campus. Each instructor in the college has a mandatory reporting responsibility related to their role as a faculty member. Faculty members are required to share information with the College when they learn of conduct that violates our Nondiscrimination Policy or information about a crime that may have occurred on Wheaton College's campus. Confidential resources available to students include Confidential Advisors, the Counseling Center, Student Health Services, and the Chaplain's Office. More information on these resources and College Policies is available at `http://www.wheaton.edu/equityandtitleIX`.

WRITING CENTER. The Writing Center is a free resource that equips undergraduate and graduate students across the disciplines to develop effective writing skills and processes. This academic year, the Writing Center is offering in-person consultations in our Center in Buswell Library, as well as synchronous video consultations online. Make a one-on-one appointment with a writing consultant here [`https://wheaton.mywconline.com/`].