

YORK UNIVERSITY

Lassonde School of Engineering

Department of Electrical Engineering & Computer Science

EECS 1028M 3.0 — *Discrete Mathematics for Engineers*

Course Outline — Winter 2022

First day of class: January 10, 2022

Instructor	Classes	Tutorials
Dr. George Tourlakis	MWF: 13:30 60 ACW 206	Mon. 14:30/120min, ACE 003
Office: LAS 2051		Mon. 19:30/120min, ACE002
e-mail: gt@cse.yorku.ca		Fri. 15:30/120min, ACE002
		Fri. 15:30/120min, ACE003
		Mon. 14:30/120min, ACE005

The tutorials are mandatory and may introduce additional examinable topics.

Students MUST attend the tutorial in which they are registered.

The first class is on Jan. 10, 2022 (eClass by ZOOM) and the first tutorial is in the week of Jan. 17, 2022.



On January 31, 2022, the current University Plan calls for classes and tutorials to transfer to in-person mode.



Reading Week (no classes, no tutorials) is Feb. 19–25.

Last date to drop a Winter 2022 (3-credit) course without receiving a grade is March 18, 2022. If you miss this deadline please note that the **Course Withdrawal Period** (i.e., ability to withdraw from a course and receive a grade of “W” on transcript) is March 19 – April 10.

This, as the title suggests, is a *first course* in discrete mathematics. That is, the kind of mathematics that people do when they do not want to be bothered with notions of *continuity* and *limits*, the latter being the major concepts and tools in the mathematics of the real and complex numbers (*analysis* or *calculus*).

Here is what we will cover:

- (1) (*Elementary*) **set theory**, which is the basic language that all *users* of mathematics “speak” and “write” (and that includes *computer scientists and engineers* —note, for example, any of the papers, old and new, that have been published over the years in the *Journal of the Association for Computing Machinery*).

We will include here

- Basic definitions and notation (e.g., empty set, unions, intersections, cartesian products).
- An *introduction* to relations and functions and their basic operations and properties.
- An *introduction* to order, equivalence relations and equivalence classes.
- An introduction to cardinality (meaning, quantifying the size of sets) and diagonalisation.
- The *Why and How* of mathematical induction as a tool to prove properties of natural numbers —in proofs **and** “constructions”.
- It is a fact of life that many objects in mathematics, but also in computer science including its applications to engineering are *defined inductively* —or *recursively* as the modern literature prefers to say.

Examples of such objects are formulas, terms, and proofs (in math and in logic), lists, trees, and entire programming languages (in computer science).

In turn, and quite naturally, *properties* of these objects are *argued by induction*.

Thus we will study inductive definitions and *how induction can be employed to prove properties of the objects so defined*. This process will persist and get enriched/refined throughout the course.

(2) **The basic vocabulary and techniques of logic**, including

- propositional calculus, and
- the elementary aspects of predicate calculus.

Here the aim is for us to thoroughly understand the tools for reasoning that the scientist[†] uses (for example, what makes “proof by contradiction” tick?)

(3) **Elementary aspects** of number theory such as

- divisibility
- “floors” and “ceilings”
- primes
- *greatest common divisor* and the “extended Euclid’s algorithm”
- notation systems for the integers

These topics prepare the student for work in the analysis of algorithms, as this subject area is dispersed in many sequel courses, and is also found in “concentrated form” in EECS 3101 3.0.

[†]All engineers are scientists, but not the other way around.

- (4) • “Big-O”-notation
- An *introduction* to recurrence relations, along with techniques to solve them, such as the method of *generating functions*.
This topic prepares for work in the analysis of algorithms, in particular it relates to the technique of “divide and conquer” of which we will see at least two instances (binary search, two-way-merge-sort).
- (5) The study of sets, relations and functions will be revisited at a more in-depth manner, including topics such as:
- Closure of relations (transitive closure, reflexive closure, etc.)
 - Inverse relations and composition of relations and functions.
 - Inductively (recursively) defined *sets* (e.g., sets of formulas of logic, sets of trees) as *closures*.
 - Structural Induction.
 - Rooted trees.
 - Counting trees that have certain properties and the relevance to algorithm analysis.

Prerequisite

MHF4U and MCV4U.

You will *not* need to know any computer programming, but, of course, a broader background always helps!

Course work and assessment:

There are **mandatory** 2-hour long tutorials for this course (**once a week**). See at the top of this outline the schedule that *pertains to the tutorial you registered in*.

(A) **Homework:** *At least 4 Problem Sets* 25%

The homework must be each individual’s own work. While consultations with the instructor, tutor, and among students, *are part of the learning process and are encouraged*, **nevertheless**, at the end of all this consultation **each student will have to produce an individual report rather than a copy** (full or partial) of somebody else’s report.



The concept of “late assignments” does not exist in this course (because full solutions are posted on the due date).



Policies.

It is important to follow these links to familiarise yourselves with Senate's and Lassonde School's *Policies and expectations* regarding *Academic Honesty*, and *Academic Integrity*.

Please also familiarise yourselves with these Senate Policies:

Academic Accommodation for Students with Disabilities, *Religious Observance* (accommodation), *Repeating Passed or Failed Courses for Academic Credit*.

But also check these two links! *Student Rights and Responsibilities* and *Student Accessibility Services*.

(B) **Mid-Term Exam** (on line): 30%

Note Date/Time: Wednesday, March 9, 2022, 13:30-14:30.



Missed tests with good reason (normally medical, and well documented) will have their *weight transferred* to the final exam. There are no “make up” tests. Tests missed for no acceptable reason are deemed to have been written and failed and are graded “0” (F). There are no “make up” assignments. **The only time** the weight of an assessed component is transferred to the final is when the component is **missed** with due cause (illness). This does not apply to assignments since the student has typically 2–3 weeks to do any given assignment.



(C) **Final Exam** during the University's Exam period. 45%

Textbook.

I know of no textbook for this course that does not contain significant errors of mathematical fact.

Therefore our reference will be my typeset lecture notes (PDF) on the main course page here:

<http://www.cse.yorku.ca/~gt/courses/EECS1028W22/safe-sets.pdf>

All chapters in our posted lecture notes support learning the material covered in class and the *learning objectives* of the course:

Course Learning Objectives (CLO)s: On successful completion of the course, students are expected to:

CLO1. Argue successfully that a collection is a set or that a relation between collections is impossible by using set-construction-by-stages principles

CLO2. Prove equality or subset relationships between sets using the assume the antecedent approach as needed

- CLO3.** Demonstrate a working knowledge via applications of transitivity, reflexivity, irreflexivity, symmetry, antisymmetry and relational transitive closure.
- CLO4.** Demonstrate a working knowledge via applications of equivalence relations and equivalence classes
- CLO5.** Demonstrate a working knowledge via applications of functional and relational composition and inverses, 1-1ness, totalness and onto-ness.
- CLO6.** Demonstrate a working knowledge via applications of finiteness, infiniteness of sets, 1-1 correspondence, enumerability and countability.
- CLO7.** Demonstrate a working knowledge via applications of proofs in Boolean logic via truth tables and in predicate logic via removing and re-inserting quantifiers and assuming the antecedent.
- CLO8.** Demonstrate a working knowledge of induction over natural numbers or inductively defined sets by proving equalities, inequalities, and more complex relations about numbers or strings
- CLO9.** Demonstrate a working knowledge via applications of partial order and Hasse diagrams
- CLO10.** Demonstrate a working knowledge via applications of big-O notation
- CLO11.** Demonstrate a working knowledge via applications of solving linear recurrence relations