YORK UNIVERSITY

Lassonde School of Engineering

Department of Electrical Engineering & Computer Science

EECS 1028M 3.0 —*Discrete Mathematics for Engineers*

Course Outline —Winter 2020

First day of class: January 8, 2020

Instructor	Classes	Tutorials
Dr. George Tourlakis	W 13:30 60 ACW 206	M 14:30 120 ACE 007
Office: LAS 2051	F 13:30 120 ACW 206	M 19:30 120 PSE 321
e-mail: gt@cse.yorku.ca		F 15:30 120 CB 115
		F 15:30 120 BC 215
		M 14:30 120 SC 303

The tutorials are mandatory and may introduce additional examinable topics.
***Students will attend the tutorial in which they are registered*.**

The first class is on Jan. 8, 2020 and the first tutorial is in the week of Jan. 13, 2020.

Reading Week (*no classes, no tutorials*) is Feb. 15–21.

Last date to drop a Winter 2020 (3-credit) course without receiving a grade is March 13, 2020. 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 14 – April 5.

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 20%

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* (acommodation), *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** (in-class): 40%

Note Date/Time: Friday, March 13, 2020, 13:30-15:20.

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. 40%

Textbook.

Kenneth H. Rosen. *Discrete Mathematics and Its Applications, Eighth Edition.* McGraw Hill, 2018.

The following chapters from the text support learning the material covered in class:
Ch. 1–2, Sections 3.3 and onwards in Ch. 3, Ch. 4, 8, 9, 11.

Course Learning Objectives: Students are expected to:

CLO1. Prove or disprove any propositional formula as the case requires, using truth tables or syntactic proof techniques such as resolution.

CLO2. Prove or disprove as the case may be simple formulas in quantified logic.

CLO3. Translate English mathematical statements into predicate logic formulas.

CLO4. Prove simple mathematical statements by contradiction, by cases, or by assuming the antecedent.

CLO5. Prove by induction statements that depend on a natural number; in particular: Prove the correctness of single loop programs and simple recursive programs.

CLO6. Prove statements about inductively defined objects by structural induction; in particular: Prove the correctness of simple recursive programs.

CLO7. Be able to reason about graphs and (binary) trees and use them in several examples, e.g., demonstrate an ability to locate the fundamental cycles of an electrical circuit

CLO8. Be able to show simple properties of trees (examples: relation between number of nodes and edges; relation between number of nodes and height)