DEPARTMENT OF COMPUTER SCIENCE and SOFTWARE ENGINEERING

Mathematics for Computer Science COMP 232

Course Outline

Fall 2017

This course is taught in five sections. The material covered in each section is the same and there are common assignments and a common final exam. The instructor may vary the order in which the material is presented to accommodate class progress. Instructors administer midterm exams independently.
This document gives important technical data about the course, which may be subject to change during the semester. Information about the course, assignments, important deadlines, and updates, can be found on the common COMP 232 web-page:

`https://users.encs.concordia.ca/~c232_2`

from where there is a link to the web-page of your section. Please consult the web-pages regularly for updates.
The instructor is the principal authority on all matters related to the course. You should take your questions to your instructor first.

Instructor	**Section**	**Office**	**e-mail**
Sabine Bergler	DD	EV3.283	`bergler@cse.concordia.ca`
Gösta Grahne	Q	EV3.109	`grahne@cs.concordia.ca`
Hovhannes Harutyunyan	R	EV3.155	`haruty@cs.concordia.ca`
Suraj Joshi	S	EV3.276	`suraj.joshi@concordia.ca`
Adam Krzyżak	PP	EV3.279	`krzyzak@cs.concordia.ca`

Course Prerequisites

MATH 203 or 209 or CEGEP Mathematics 103, previously or concurrently.
MATH 204 or 208 or CEGEP Mathematics 105, previously or concurrently.

Course Description

Propositional logic and predicate calculus. Sets. Functions and relations. Elements of number theory. Proof techniques: direct proof, indirect proof, proof by contradiction, existence proof. Recursive definitions and inductive proofs. Equivalence relations and partial orderings.

Course Learning Objectives

Introduce students to the basic abstractions from Discrete Mathematics that are of central relevance in Computer Science. Teach students to reason formally using these abstractions, and to recognize and apply them in various areas of Computer Science and Software Engineering. Prepare students for courses on the foundations of computation.

Course Learning Outcomes

Upon successful completion of the course students will have basic knowledge and skills in mathematical and formal reasoning. In particular, students will be able to

- apply propositional logic, truth tables, logical inference, predicate logic and quantification as tools to describe formal objects and their properties
- use proof techniques, inductive proofs and recursive definitions to reason about formal objects
- understand the concepts and properties of sets, functions and relations, and use them to describe discrete objects
- carry out elementary calculations in modular arithmetic and understand its use in computer systems

Textbook

COMP 232 Custom publication for Concordia University: Selected chapters from *Discrete Mathematics and its Applications*, 7th edition, by Kenneth Rosen, McGraw-Hill, New York 2012.

Attendance

Students are responsible for all material presented in lectures and tutorials.

Tutorials

This course has a scheduled tutorial, which is an integral part of this course. It consists of discussion of problems given by the tutor or suggested by the students. Tutorials provide time for students to solve exercises with immediate feedback; active participation is therefore vital for students' progress.

Assignments

There will be four assignments that will be posted on the common course web page. The instructor will announce a time and place to submit your solutions. Late assignments will not be accepted.

While discussion of the assigned problems among students is encouraged, each student must solve the assignment problems independently. Students should be aware of the University's Code of Conduct (Section 17.10.3 of the Undergraduate Calendar) concerning cheating, plagiarism, and the possible consequences of violating this code. A signed *Expectations of Originality* form, available from the course website, must be completed, signed, and submitted to the instructor *one time only* at the beginning of the term, and no later than the second week of classes.

Solutions to assignments must start with the student's name and I.D. number, the course number and section number, the instructor's name, the assignment number, and the date of submission. Furthermore, on each submitted assignment you must write the following statement: "*I certify that this submission is my original work and meets the Faculty's Expectations of Originality* ", together with your signature.

Problems in the assignments will be graded on the following basis: a correct answer gets 100%, a reasonable attempt gets 50%, and no attempt or a very poor attempt gets 0%. Only a subset of the assignment problems will graded; however, solutions will be posted shortly after the due date.

Term Test and Final Exam

There will be one term test, which contributes 30% to the final grade. There will be a common three-hour final examination during the examination period. No tools are allowed during the term test or the final exam; in particular no textbooks, no crib sheets, and no calculators. The final exam will cover material from the entire course.

Weight Distribution

Assignments	10%
Term Test	30%
Final Exam	60%

If the mark for the final exam (as a percentage out of 100) is higher than the term test mark (as a percentage out of 100) then the weight of the term test will be shifted to the final exam. Thus students will benefit from better performance on the final exam.
To pass the course, the student must have a passing mark on the final and on the term test as well as a passing total score. There is no standard relationship between numerical percentages and the final letter grades.
In the event of extraordinary circumstances beyond the University's control, the content and/or evaluation scheme in this course is subject to change.

Topics

The approximate timetable is shown below. Students will benefit greatly by reading the relevant sections of the textbook and/or the lecture notes *before* coming to class.

Week	Topics	Assignment	Sections
1	Propositional Logic		1.1, 1.2, 1.3
2	Predicates and Quantifiers	1	1.4
3	Nested Quantifiers		1.5
4	Methods of Proof	2	1.6
5	Proof strategy		1.7, 1.8
6	Proofs, Sets		2.1, 2.2
7	Functions, Cardinality of sets		2.3, 2.5
8	Elements of number theory	3	4.1
9	Number theory		4.3
10	Mathematical Induction		5.1, 5.2
11	Recursive definitions, Relations	4	5.3, 9.1, 9.3
12	Closures of Relations		9.4
13	Equivalence Relations, Partial Orderings		9.5, 9.6

CEAB Graduate Attributes

As part of either the Computer Science or Software Engineering program curriculum, the content of this course includes material and exercises related to the teaching and evaluation of graduate attributes. Graduate attributes are skills that have been identified by the Canadian Engineering Accreditation Board (CEAB) and the Canadian Information Processing Society (CIPS) as being central to the formation of Engineers, computer scientists and information technology professionals. As such, the accreditation criteria for the Software Engineering and Computer Science programs dictate that graduate attributes are taught and evaluated as part of the courses. The following is the list of graduate attributes covered in this course, along with a description of how these attributes are incorporated in the course.

Attribute 1: Knowledge-base: Knowledge of sets, propositional logic and predicate calculus, functions and relations, elements of number theory. Proof techniques: direct proof, indirect proof, proof by contradiction, proof by induction.

Indicator 1.1: Knowledge base of mathematics: Demonstrate knowledge of: Sets, Propositional logic and predicate calculus, Functions and relations, Number theory, Proof techniques: direct, indirect, contradiction, induction.

Attribute 2: Problem analysis: Use mathematical knowledge and proof techniques to analyze problems related to computer and software systems.

Indicator 2.1: Problem identification and formulation: Identify and correctly formulate all different parts of the problem. Understand how the various pieces of the problem relate to each other and the whole. Identify parts of the problem that may affect the development of the solution. Analyze and take into consideration the operational context of the problem.

Indicator 2.3: Problem Solving: Extract relevant parameters, assumptions and variables from the problem statement. Use logic and other formalisms to formulate a model of the problem. Develop analysis models and diagrams to express the problem.