section 0301

CSC236: Introduction to the Theory of Computation

(Dustin)|(Dr. Wehr)|((Instructor |Professor)(Wehr)?)
wehr@cs.toronto.edu - prefix subject line with [CSC236]
Office: SF 4306 D

Prerequisites: material covered in CSC165 and CSC148

Resources:

- Today's slides: www.cs.toronto.edu/~wehr/236/slides_week1_6jan.pdf
- Course webpage on Piazza: https://piazza.com/utoronto.ca/winter2015/csc236h1/home
- Free book: http://www.cs.toronto.edu/~vassos/b36-notes/notes.pdf
- Help Centre (**I'll be there 2 days per week**):
  4-6pm Mon-Thurs, Bahen building room 2230
  Starts next week.
- My and Gary's office hours: we'll tell you soon.
  → I'm available to talk after class today.
- Lecture/tutorial times: http://coursefinder.utoronto.ca/course-search/search/courseInquiry?methodToCall=start&viewId=CourseDetails-InquiryView&courseId=CSC236H1S20151

What’s this course about and why do we make you take it?

- Mainly about: mathematical problem solving that's useful for computer scientists

Why we make you take this course

(a sample of the reasons)

(1)

- Same reason engineers have to take physics.

- Absolutely necessary for every programmer?

- No

- Will you need this to be a *confident programmer*?
  = programmer who knows that he or she can write **any** software given enough time and sufficiently clear requirements.

- Almost certainly

Why we make you take this course

(a sample of the reasons)

(2)

- Most of you are bad at explaining your code (what it does and how it does it).
- This course will make you better.
- If you don't get better, you'll have issues when you're not the only person working on a software project (other programmers won’t want to work with you).

Topics: Deductive reasoning about programs

- Deductive ≣ reasoning that’s **potentially presentable** as a proof
- Especially proofs by (various flavours of) induction
- Equivalence of the different forms of induction
- Proving the correctness of programs
- More asymptotic runtime analysis

Topics: Foundations of Theoretical Computer Science

- Formal language theory - the mathematical study of *languages* = sets of strings.

- Special focus on *regular languages*

  - Suffice for a lot of complex text searching and manipulation tasks.

  - Their basic properties

  - Their computation models

  - Title slide contains a *regular expression* (one of the computation models) that defines the set of strings:
    {“Dustin”, “Dr Wehr”, “Instructor”, “Professor”, “Instructor Wehr”, “Professor Wehr”}

Why we make you take this course

(a sample of the reasons)

(3)

- As a working programmer, will you ever have to explicitly prove one of your programs correct?
- Probably not.
- But you *will* sometimes *sketch* the idea of proof, on paper, to convince yourself that you *could* prove your program correct.
- You will *often* use deductive reasoning in your head to
    - design algorithms
    - adapt well-known algorithms for your purposes
    - optimize your code
    - debug your code

Today: Induction on the chalkboard

- Notes for today from 2011 by Gary: http://www.cs.toronto.edu/~wehr/236/week1/CSC236.2011W.Week-01.pdf

- Slides by Danny Heap containing the examples we'll cover today: www.cs.toronto.edu/~wehr/236/week1/danny_2011_lecture1.pdf

- **NOTE:** in the future, all lecture/note materials will be posted on Piazza here: https://piazza.com/utoronto.ca/winter2015/csc236h1/resources