Introduction to the Theory of Computing
CS 381 - Fall 2015
Credits: 3 hours

Instructor: David Furcy **Office:** Halsey 221
Email: furcyd@uwosh.edu **Phone:** 424-1182
Office Hours: **MWF** 12:30-1:30, **T** 10:00-noon, **R** 11:00-noon, or **by appointment**

Class Meetings: **MWF** 9:10-10:10 in Halsey 456

Prerequisites: CS271 and either Math-212 or Math-222, both with a grade of C or better.

Class Web Page: On D2L

Required Text: *Introduction to Computer Theory* (second edition)
-- Daniel I. A. Cohen (J. Wiley & Sons, Publisher)

Tests: Exam #1: Week of October 12th
Exam #2: Week of November 9th
Exam #3: Week of December 14th

If you have special needs, please come and talk to me at the end of the first class so I can accommodate your needs right away.

Course Overview: As part of a liberal education, this course brings the rigor and formalism of mathematics to bear on philosophical and practical questions related to the complexity and even the limits of computation. We will analyze problem solving and computation from a language-processing perspective. We will discuss fundamental results in theoretical computer science and interpret them in the context of real-world computing. We will be concerned, for example, with answers to the following questions:

- What, in the abstract, is a *computing machine*?
- When are two such machines equivalent in computing power?
- What does it mean for a problem to be *computable*?
- Which problems are/are not computable and how can we tell them apart?
- What is an *algorithm*?
- What is the difference between languages and machines?

Course Outcomes:

1. Given a language recognizer and a string, the student will be able to decide whether or not the string is accepted by the recognizer.
2. Given an informal description of a language, the student will be able to categorize it (as a regular, context-free, recursively enumerable, etc. language) and to prove the correctness of this categorization.
3. Given an informal description of a language, the student will be able to build a computational model to recognize it.
4. Given a computational model and an alphabet, the student will be able to describe the language that it recognizes or generates.

5. Given a computational model, the student will be able to transform it into an equivalent computational model of a different type (for example, transforming a regular expression into a transition graph, or a context-free grammar into a pushdown automaton).
6. Given two regular languages, the student will be able to design a computational model that recognizes their intersection and union.
7. Given a formal language and a set of languages, the student will be able to apply the appropriate pumping lemma or a known closure property to prove that the language is not a member of the given set.
8. Given a formal grammar, the student will be able to judge whether it is ambiguous or not.
9. Given an ambiguous grammar for a language that is not inherently ambiguous, the student will be able to eliminate the ambiguity.
10. Given a formal grammar and a set of requirements pertaining to the allowed format of productions (for example, whether lambda or unit productions are allowed, or whether the grammar should be in Chomsky normal form), the student will be able to transform the given grammar into an equivalent grammar that meets the format requirements.
11. Given a regular or context-free language formally specified by a computational model, the student will be able to solve the finiteness and emptiness problems for this language.
12. Given a context-free grammar, the student will be able to generate the set of useless non-terminals and productions in the grammar.
13. Given a context-free grammar and a string, the student will be able to trace the CYK algorithm to solve the membership problem.
14. Given a deterministic Turing machine, the student will be able to encode it into a string (and vice-versa), and determine whether the machine accepts its own encoding.

Topic Coverage: We will cover the following topics:

- Recursive definitions
- Regular expressions, finite automata (DFAs, TGs, NFAs), Kleene's theorem
- Context-free grammars, pushdown automata
- Regular and context-free languages and corresponding pumping lemmas
- Turing machines
- Recursive and recursively enumerable languages
- Church's thesis
- Decidability and algorithms

Course Grading Policy: Your overall grade for this course will depend on pop quizzes, homework assignments, and exams. All assignments (quizzes, exams, respectively) will carry the same weight when computing your overall assignment (quiz, exam, respectively) grade. Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

Component	Weight
Quizzes	15%
Assignments	25%
Exams	60%

Your final letter grade for the course will be computed using the following mapping:

Numerical Score	Grade	Numerical Score	Grade
≥ 92	A	≥ 72	C
≥ 90	A-	≥ 70	C-
≥ 88	B+	≥ 68	D+
≥ 82	B	≥ 62	D
≥ 80	B-	≥ 60	D-
≥ 78	C+	<60	F

I will be glad to discuss any questions you may have about grades. However, make sure to bring them up right away, upon return of each graded assignment or exam. Last minute requests, especially after the final exam, will not be entertained.

Attendance and Participation: You are expected to not only attend **every** class meeting but also to come **prepared** for and **participate** actively in it. Necessary preparation requires you to have studied and assimilated the material covered in previous sessions, to have met with me outside of class to discuss any questions you may have, to have done the assigned reading, and to have completed the assignments on time.

It is hard to imagine how a student could do well in this course while missing classes, attending them unprepared, or not participating.

On the positive side, I have high expectations for my students and will always support and encourage you. I **strongly encourage** you to **ask any question** or raise any issue you have with the course either during or at the end of class, or during my office hours. I will also gladly meet with you by appointment. Send me email or give me a call to make an appointment. While I will meet with you as soon as my schedule permits, do not expect me to be widely available before an assignment is due.

Late Submissions: I will describe the submission procedure for your assignments when the time comes. However, let me point out right away that each assignment will come with a deadline (day and time) after which any submission is considered late, **with no exception**. The late-submission policy works as follows. If your submission is past the deadline on the due date, you will lose 10% of your score. If your submission reaches me 1, 2, 3, or 4 days after the due date, I will grade your assignment as if it were turned in on time and then reduce your numerical grade by 20%, 40%, 60%, and 80%, respectively. If your submission is more than 4 days late, I will not grade it and you will receive a zero. Late submissions can easily be avoided by starting to work on the assignment right away and asking me questions early if you get stuck.

The penalty for late submissions can be waived in **only one** scenario, namely if you give me a signed note from a doctor or a written justification for the extension from the Dean of Students Office. If you miss an exam, you **may** be able to take a make-up exam provided you give me a valid justification (see above) ahead of time if possible. Only one make-up exam will be given. It will be a comprehensive exam scheduled at the end of the semester. If you miss a quiz, you **may** be able to take a make-up quiz, provided you have a valid justification (in writing) for your absence.

Collaboration versus Cheating: All submissions must be the work of only one student, namely the one whose name appears on the submission. While it is acceptable and encouraged to discuss

the assignments with others, you must submit your own work unless you can live with a zero and the other potential academic sanctions of cheating. Check out the UWO Student Discipline Code (UWS 14) at http://www.uwosh.edu/stuaff/images/student-discipline-code/ for details.

In conclusion, remember that computer science classes require a lot of work in addition to active participation in class. It takes considerable practice to develop the technical and analytical skills targeted by this course. You will need to spend **at least, and typically much more than, three hours of effort outside of class for each in-class hour**. Having said this, I expect every hardworking student to do well in this course.

Have fun this semester and good luck!