COS 435
Theory of Computation

Department of Computer Science and Engineering
Taylor University

Three Credit Hours
Euler Science 200 • MWF 11:00–11:50

Final Exam: M/13-Dec, 8:00–10:00AM

1 Instructor

Dr. Josh Hibschman
Adjunct Professor, Computer Science and Engineering

Office	Euler Science Complex 208
Email	`josh_hibschman@taylor.edu`
Hours	MWF 9:00–12:00
	Or by appointment

2 Description

Prerequisites: COS 265 or Related Proficiency in Algorithms

From the catalog: Theory of computation (TOC) is a theoretical treatment of what can be computed and how efficiently computation can be done. Topics include models of computation and automata, deterministic and nondeterministic computations, and formal language theory.

TOC is a branch of Computer Science that is concerned with how problems can be solved using algorithms, how efficiently they can be solved, how hard they are to solve, and how hard it is to verify whether they were solved. Real-world computers perform computations that by nature run like mathematical models to solve problems in systematic ways. This course will provide an introduction to TOC through the following teaching methods.

1. Applying the principles, processes, and models of computation in independent projects,
2. Discovering both how to generalize and identify limitations for computational models from reading and problems,
3. Exploring student-driven design in a hands-on group project,

4. Exploring applications and tradeoffs between theoretical models,
5. Creating and evaluating prototypes of advanced models,

This course is an elective available to all students in computer science and is an option in systems. According to the standard policy, a grade of C- or better must be obtained in this course to meet the COS major or systems requirement.

3 Learning Objectives

Upon successful completion of this course, you will be able to:

1. explain computation in terms of automata,
2. clearly define the boundaries of computation,
3. recognize complexity,
4. adapt theoretical techniques to real-world applications,
5. distinguish tradeoffs in current tools – limited by theoretical boundaries, and
6. find reasonable workarounds when confronted by the halting problem.

4 Course Outline

We will cover the following topics.

1. DFA: Deterministic Finite Automata
2. NFA: Nondeterministic Finite Automata
3. Regex: Regular Expressions
4. Pumping Lemma
5. CFL: Context Free Languages
6. CFG: Context Free Grammars
7. Pumping Lemma (Context-Free)*
8. NPDA: Nondeterministic Pushdown Automata
9. DPDA: Deterministic Pushdown Automata*
10. DCFG: Deterministic Context Free Grammars*

11. LBA: Linear-Bound Automata*
12. TM: Turing Machines
13. Recognizability
14. Diagonalization*
15. Decidability
16. Reducibility
17. Time Complexity (w/ review of Big-O)
18. P vs NP
19. NP Hard, EXPTime*
20. Space Complexity*
21. Independent Exploration of Advanced Algorithms*

*Although we will spend some discussion time on the topics, this course will not cover all topics in depth. Some topics marked above may be covered less thoroughly than the core topics.

We will wave our hands at the following topics throughout the course:

1. programming language grammars
2. compilers
3. distributed computing
4. quantum computing
5. quantum turing machine variants
6. cellular automata
7. probabilistic algorithms
8. number theory
9. machine learning, markov chains

5 Team Project

Each student will participate on a team for the team project towards the end of the semester. The team project involves independent study of a theoretical model not discussed in the course, building a prototype, and demonstrating the effectiveness of the model in an in-class demo. To receive full credit, the following requirements must be met.

1. Proper computational model applied.
2. An appropriate real-world application is described.
3. Without the model, solving the problem would be more complex.
4. Real-world problem applies to some relevant issue today.
5. Students must find a way to improve upon or add to the model.

Team Project scores will not depend on how clever, elaborate, or even successful the final product is. Instead, the teams will be assessed based on how well they apply the principles above and what they learn through the process.

Any project idea must be explicitly approved by me by the due date set in the course Canvas page. Teams will pitch their idea to the professor. Each team should expect their project pitch to be critiqued, so come prepared. Good project ideas solve an actual (not perceived) problem using a physical device (not web page or mobile app).

All teams will present their project, results, and learning points during the last week of classes and during the final exam period.

Each student will turn in a team evaluation and an evaluation for each team member, including a self- evaluation. This evaluation will inform me on how to grade participation.

6 Final Presentation

This course has no final exam. Instead, each student will present a final project. Details for the project will be posted early in the course and discussed early April, and the project is due by the final exam period.

7 Participation

Students will have many opportunities to participate throughout the course, inside and outside the classroom. I will assess performance using many measures, such as number of in-class questions, answers, and discussions, daily reading assignments, quality and quantity of office hours visits. Student participation will also depend on how involved they were in the team project.

8 Required Textbooks

There is one required textbook for the course.

1. (TOC) Introduction to the Theory of Computation, 3rd edition, Michael Sipser, Cengage Learning, 2012. ISBN 978-1133187790.

The TOC book introduces models and proofs describing computation in terms of Automata, Computability, and Complexity.

Lectures are based around the assigned readings, providing more examples and time of discussion. I expect students have read the assigned readings before the corresponding lecture so that we can have a meaningful discussion on the topic.

9 Golang

Students will be required to complete implementations of various computational models and problems in Golang. No prior experience is necessary, however students should have had significant experience in Python, Java, JS, C++ or related prior to taking the course.

10 Classroom Atmosphere

This classroom experience is designed to enable students to learn from course material, the professor, hands-on experience, and the work of other students. As a learning community, we should come to class prepared to discuss readings and critically evaluate project experiences.

Students will be working in teams. Teams provide diversity of thought, and this diversity sometimes creates tension. Students should be aware that this tension is sometimes the very ingredient that enables the most creative ideas to be born. When tension arises, students are encouraged to reflect on the diversity of the team and lovingly and respectfully seek to work through the issues. I will be available to advise any team on group dynamics and collaboration.

11 Evaluation

The grading breakdown for the course is shown in Table 1. Refer to my *Periodic Table of the Grades* for the grading scheme. I reserve the right to award a higher grade than strictly earned; outstanding attendance and class participation figure prominently in such decisions.

Category	Weight
Assignments	40%
Projects	40%
Participation	20%
Total	100%

Table 1: Grading details

12 General Course Policies

The following are general policies and details for courses that I instruct. Unless otherwise stated above, these policies apply to this course. Please read them carefully.

12.1 Reading Response Scoring and Grading

Assignments are broken down into individually graded items. Any item to be graded will be scored on the 3-point scale shown below.

1. 0: unsatisfactory, nothing of value was submitted, does not satisfy requirements
2. 1: solution submitted, but clearly not correct, contains serious flaws
3. 2: solution is satisfactory, demonstrates understanding but with minor flaws, only partially correct
4. 3: perfect solution, meets every requirement and expectation, clearly demonstrates thorough understanding

A score of 0, 1, and 3 are clearly identifiable; anything not clearly identifiable is scored a 2.

The reason for using discrete values is to avoid arguments over unimportant issues, to remove (as much as possible) the subjectivity in grading, to allow (as much as possible) room for unit/automatic testing systems, and to reduce the turnaround time for receiving a grade. Furthermore, this scoring can often be more telling than some arbitrary number of points.

I convert an individual item score to a letter grade as follows: 0 maps to an F, 1 to D-, 2 to C (average), and 3 to A. For a final grade for an assignment.

Note: there will be no rounding up when determining your final score. I do reserve the right to award a higher grade than strictly earned; outstanding attendance and class participation figure prominently in such decisions.

Exams and final course grade will be scored similarly.

I have based my grading philosophy on an article by Dr. William J. Rapaport: http://www.cse.buffalo.edu/ rapaport/howigrade.html

13 Late Assignments

Late work is not accepted.

As in the world outside academia, time management is a valuable and import skill to master. The assignments for this course are designed to help you learn and master the computer science material. As in the world outside academia, time management is a valuable and important skill. In an effort to help you understand and master the material as well as to help you develop good time management skills, no assignments will receive credit if turned in late. Moodle assignments will have a cut-off set for the due date and time, and only commits with timestamps before the due date and time will be accepted for Git-based assignments.

14 Course Expectations

Following are my expectations regarding the course.

14.1 Attendance

It is my highest goal to provide opportunities for authentic learning to take place in the classroom. This means that lectures should be personally meaningful, teaching should engage the students to think in multiple modes of the discipline, and the activities motivate the students to engage in self-driven learning.

If a student is physically, emotionally, mentally, or spiritually unwell I urge the student to skip my class and focus on getting better.

If a student does miss class, the lectures will be recorded and made available online. It is important to stay caught as the work builds upon itself.

14.2 Late Work

All course assignments will include an unambiguous due date. Usually, assignments are due in the evening after class on the due date. Barring exceptional circumstances like those mentioned in section 14.1, I expect your work to be submitted *on the due date*. Late work will *not* be accepted.

This policy on late work is intended to prepare you for real-world experience after graduation. In the marketplace, late work is not merely an inconvenience. Missing a deadline may alienate your customer, upset your manager, ruin your project, or terminate your employment! *Now* is the time to learn the self discipline and time management skills required to complete your work when it is due.

14.3 Conduct

I expect you to be prepared, awake, aware, and participatory during class. I will not hesitate to ask you to stand or move if you are distracted or sleepy.

I expect you to join in discussions, respond to questions from me and from your colleagues, and ask questions of me. I expect you to hold my feet to the fire if I am being unclear, unkind, or contradictory.

14.4 Computer Use

Students are given opportunities to practice self-regulation in the classroom. I encourage device usage during class, as long as that usage contributes to learning. Bring your devices. Use them. Be prepared to share whatever is on your screen at any arbitrary point in time.

15 Pandemic

We will adhere to university guidance regarding the COVID-19 pandemic.

> Taylor University is committed to its academic and spiritual mission, even as we teach and learn in a global pandemic. In the interest of your safety, and the health and wellness of others in the classroom, campus, and community, you can expect to have social distancing and precautionary measures used in the classroom. **This means you are required to wear your own face mask in class**. You may also be asked to remain seated at a distance away from other students when you are in the classroom. Some larger classes may create discussion groups and establish a rotation for some students to join the class using virtual tools. As students, you can expect that your professor will communicate clearly and regularly about expectations for this course. If course delivery methods need to change or adapt to health concerns, updates will be announced in class sessions.

16 Course Management

We use several systems to help manage the course and for on-line communication.

16.1 Email

Electronic mail is an official channel of communication between all members of the university community. You are responsible to check your email regularly (daily) for information related to the course.

16.2 Canvas

The Computer Science and Engineering department uses Canvas as our Learning Management System. The URL for Canvas is `https://canvas.cse.taylor.edu`.

You are responsible for checking Canvas regularly to keep up with assignment due dates and other announcements. For due dates, *the Canvas calendar is your friend.*

16.3 Slack

This course will use Slack for informal communication, Q&A, last minute announcements, jokes, and the like. Find the *TU CSE Student* slack team at `tucsestudents.slack.com`. Look there for a *channel* dedicated to the course.

17 Final Exam Policy

Students must take their final examinations at the assigned hours listed in the schedule of classes. Exceptions will be made only because of serious illness or the death of an immediate member of the family. Reasons such as plane schedules, availability of flights, and rides leaving early are not acceptable exceptions. Students having more than two examinations on the same should report this to the Registrar's Office *ten days prior to the beginning of finals week.* Reasonable alternatives in alleviating this dilemma will be pursued by the registrar and the student in consultation with the appropriate faculty.

18 Academic Integrity

Academic dishonesty constitutes a serious violation of academic integrity and scholarship standards at Taylor that can result in substantial penalties, at the sole discretion of the University, including but not limited to, denial of credit in a course as well as dismissal from the University. In short, a student violates academic integrity when he or she claims credit for any work not his or her own (words, ideas, answers, data, program codes, music, etc.) or when a student misrepresents any academic performance.

In an instructional setting, **plagiarism** occurs when a person presents or turns in work that includes someone else's ideas, language, or other (not common-knowledge) material without giving appropriate credit to the source. Plagiarism will not be tolerated and may result in failing this course, and may also result in further consequences as stipulated in the Taylor University Undergraduate Catalog.

For purposes of this course, the following are non-exhaustive examples of **activities that violate academic integrity**:

- Sharing code or other electronic files by copying, retyping, looking at, or supplying a copy of a file from this or a previous semester
- Sharing written assignments or exams by looking at, copying, or supplying an assignment or exam

- Looking at code from others students, from this or previous offerings of the class, from courses at other institutions, or from any other source (e.g., software found on the Internet)
  - If you find someone else's solution or partial solution to an assignment, you are you longer even able to solve the problem independently. You can't un-see the existing solution. Your solution can no longer be your own - it's your repackaging of someone else's work and is an express violation of academic integrity.

In contrast, the following are non-exhaustive examples of activities that DO NOT violate of academic integrity:

- Looking at general (not assignment solution specific) technical documentation (man pages, API references, etc)
- Using code provided by the instructor
- Clarifying ambiguities or vague points in class handouts or textbooks
- Helping others use the computer systems, networks, compilers, debuggers, profilers, or other system facilities without regard to a particular assignment or project
- Helping others with high-level design issues
- Helping others with high-level (not code-based) debugging

19 Support Services

Be aware of the following support services available to you as a Taylor student.

19.1 Academic Assistance

The Academic Enrichment Center (AEC), located in the Zondervan Library, provides individualized academic skills help (e.g. test preparation, note taking, planning, etc.). Contact **Dr. Scott Gaier**, `scgaier@taylor.edu`.

19.2 Tutoring

Peer Tutoring Services, located in the AEC in Zondervan Library, provides free help to students in most content areas. For further information, contact **Darci Nurkkala**, `drnurkkala@taylor.edu`.

19.3 Students with Special Needs

The Academic Enrichment Center provides a variety of services for students who have physical or other disabilities. If you need accommodations due to a disability, please contact **Lisa Wallace**, `lswallace@taylor.edu`.

This document last updated August 31, 2021.