CS 107: Compilers

Fall 2023

https://canvas.tufts.edu/courses/50025/

Class Sessions	Monday, Wednesday 1:30pm-2:45pm
	Anderson Wing SEC, Room 210
Instructor	Richard Townsend, richard@cs.tufts.edu
Richard's Office Hours	Tuesday 10:30am-12:30pm (Or by appointment)
	Location: JCC 440A
Teaching Fellows (TA)	Jacob Zimmerman, Jacob.Zimmerman@tufts.edu
	Ronit Sinha, Ronit.Sinha@tufts.edu
Jacob's Office Hours	Wednesday 11:00pm-1:00pm
	Location: JCC 3rd Floor Open Area
Ronit's Office Hours	Friday 12:00pm-2:00pm
	Location: JCC 3rd Floor Open Area

1 Course Overview

Summary

In CS 107, you will learn about the design and implementation of modern compilers. The course will focus on the main steps of general compilation (Scanning, Parsing, Semantic Checking, and Code Generation), while also introducing compilation techniques for language-specific features and some traditional compiler optimizations.

Outside the classroom, the focus of the course is an intensive, semester-long project: you and a team will design a small programming language and implement a compiler for it using the OCaml language. The algorithms and concepts you will learn have broad application outside of the course: many programming tasks can be understood as variations of interpretation or translation, and understanding how a compiler operates will further develop your abstract thinking skills and make you a better programmer.

Prerequisites

- CS 40 Machine Structure and Assembly-Language Programming: You will be dividing into teams to build a compiler, so you need to know how to keep a large software project under control (e.g., Makefiles and source code control systems, budgeting time, allocating work to team members). You also should have a semester's worth of C programming experience and understand how primitive types are represented at the machine level.
- CS 105 Programming Languages: You will need some experience writing code in a functional language that provides higher-order functions, first-class functions, algebraic data types, and pattern-matching. It is also assumed that you have a basic grasp of type checking.

Course Goals

By the end of this course, students should be able to

1. appraise the benefits and limitations of functional programming.
2. relate formal language theory to front-end compiler implementation.
3. describe a compiler's responsibilities and their ramifications on a computer system as a whole.
4. motivate and describe the four essential phases of a compiler.
5. recognize the difficulties faced by a group working on a large, long-term software project, and develop potential solutions to those difficulties.

2 Technical Resources

Piazza

We will be using Piazza for class discussion; we have a specific 107 piazza page that is also linked from the Syllabus page of the course Canvas site. **Piazza will host all of our major course announcements; it is your responsibility to check in regularly to avoid missing any important information.**

In general, you should post all questions related to the course on Piazza; post publicly if at all possible, as other students may have similar questions or be able to help you faster than the course staff. Please post privately to the course staff if your question reveals individual work (e.g., part of a solution to a homework problem) or concerns your grades.

Suggested Textbook(s)

While this course has no required textbook, the following references can help facilitate your understanding of the covered content.

- *Modern Compiler Implementation in ML.* Andrew W. Appel. Cambridge University Press, 2004.

  An excellent book on how to actually implement a compiler in ML (Appel also published Java- and C-based textbooks).

- *Compilers: Principles, Techniques, and Tools, 2nd Edition.* Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. Addison-Wesley, 2006.

  The "dragon book" has long been touted as the definitive textbook on the theory of compiler design. While some sections have become dated due to language evolution and improvements in computer architecture, many are still completely applicable today (especially those on front-end compiler design and compiler optimizations).

- *Engineering a Compiler, 2th Edition.* Keith D. Cooper and Linda Torczon. Morgan Kaufmann, 2012.

  A combination of the previous two textbooks, one might say. They cover not only lots of the theory of compiler design, but also suggestions on how to implement compiler passes on today's machines.

Class Materials
All slides will be posted on the course website within a day of their presentation in class. They can be found in the "Files" section of Canvas, with links to appropriate slidedecks in the "Modules" section.

OCaml resources
Start at the OCaml homepage to find compiler installation instructions and documentation and the user's manual for the whole system, including ocamllex, ocamlyacc, and all standard libraries.

OCaml tutorials: Real World OCaml is a cogent, well-written textbook (completely online) that covers the OCaml language; the first chapter is an excellent complement to my OCaml slides. The official website also has a collection of teaching problems, but these are more loosely organized.

Project resources
Some examples are provided on the course website to give you a feel of what can make a stellar project (see the second and third Modules). Feel free to use these examples for inspiration, but avoid any form of plagiarism.

The success of your project is strongly tied to your ability to use version control systems like git or svn. These tools will let you collaborate on code files, track each member's contributions and modifications to the project, and ensure everyone has access to the same codebase. The Tufts CS department provides students with free public or private Gitlab instances; take advantage of them! If you've never used git before, here are some tutorials:

- An intro to git and github for beginners.
- A collection of guides for github use.
- Full documentation on the git version control system.

We will not be covering git in class; please post on Piazza or attend Office Hours if you need assistance setting up or using a version control system.

3 Tentative Schedule

Below you will find the tentative schedule for the semester. Topics and assignments are subject to change, and may be updated as the semester progresses (you will be informed of any changes as soon as they happen).

The red highlighted class periods indicate sessions where your group will meet with your project advisor to receive feedback on important project deliverables and check-in concerning your progress. It is imperative that all group members attend each of these sessions!

Week	Date	Topic	Assignments	
			HW	Project
1	Sep 06	Course Overview and Intro. to Languages		
2	Sep 11	Language Processors and OCaml Intro		
	Sep 13	Essential OCaml I		
3	Sep 18	Essential OCaml II		
	Sep 20	Lexing I: Regular Expressions and MicroC		Proposal
4	Sep 25	Proposal Feedback		
	Sep 27	Lexing II: Theory of OCamllex	HW 1	
5	Oct 02	Parsing I: Grammars and MicroC Lab 1		
	Oct 04	Project Check-in		
	Oct 06	(Lab 1 Due Only)	Lab 1	
6	Oct 09	NO CLASS		
	Oct 11	Parsing II: Shift-Reduce Parsing		
7	Oct 16	Parsing III: Theory of OCamlyacc		Scanner & Parser
	Oct 18	Semantic Analysis and MicroC Lab 2		LRM
8	Oct 23	Code Generation I: IRs and LLVM	Lab 2	
	Oct 25	LRM Feedback	HW 2	
9	Oct 30	Code Generation II: MicroC Lab 3		
	Nov 01	Project Check-in		
10	Nov 06	Runtime Environments I: The Stack	Lab 3	
	Nov 08	Runtime Environments II: The Heap		Hello World
	Nov 10	SCHOOL HOLIDAY		
11	Nov 13	Runtime Environments III: More Heap!		
	Nov 15	Project Check-in		
12	Nov 20	Heap Conclusion and Optimization I: Intro		
	Nov 22	NO CLASS		
	Nov 23	NO CLASS		
	Nov 24	NO CLASS		
13	Nov 27	Optimization II: Classic Optimizations		
	Nov 29	Optimization III: Liveness Analysis		Extended Testsuite
14	Dec 04	Project Check-in	HW 3	
	Dec 06	Course Wrap-Up		
15	Dec 11	Buffer Class		
	Dec 12	Reading Period		
	Dec 13	Reading Period		
	Dec 14-15	Presentations		
	Dec 15	Project Report Due		

The "Buffer" classes listed near the end of the course are meant to give the course schedule some flexibility; if more time is required for some earlier concepts, the schedule can be pushed back without sacrificing later content. If the course ends up moving faster or as anticipated, the content of this class will be determined jointly by the instructor and students.

4 Assessments and Grading

Assessment will be based on five criteria:

1. One semester-long group project (70%): Design the goals, syntax, and semantics of a little language, and implement a compiler for that language. Graded deliverables are as follows: Proposal (2%), Scanner & Parser (5%), LRM (5%), Hello World (5%), Extended Testsuite (8%), Final Submission and Project Report (45%).
2. Three Homework Assignments (15%): These written assignments will gauge your understanding of theoretical compiler concepts and further prepare you for the project.
3. Three Labs (5%): These labs will take place during class sessions, giving you and a partner the chance to explore an example compiler through a guided set of questions.
4. Participation (5%): Your engagement in class sessions and contributions during group check-in and feedback sessions will be taken into consideration for this grade.
5. Peer-Assessment (5%): As part of your final project deliverable, you will privately assess your teammates' contributions to the project as a whole. Your teammates' assessments will factor into your final grade.

Homework, Lab, and Project Deliverable Logistics

All homework, lab, and project deliverables must be submitted via Canvas. No submissions will be accepted via email or other means. Project deliverables will be submitted as a group, i.e., only one member will submit the deliverable.

All deliverables are due at 11:59:00PM on their due date. To avoid missing this strict deadline, you should submit whatever you have a bit earlier even if incomplete; you may submit as many times as you wish before the deadline. We will always grade the latest submission.

Each deliverable will have specific instructions posted on Canvas. These instructions must be followed exactly to receive a full grade.

Late Policy

All students have a budget of **three "late tokens"** for any deliverable *except the proposal and the final project report.* It is your responsibility to track how many late tokens you've used over the semester. By expending a late token, you get a 24-hour extension on an assignment. This extension happens automatically: when you turn in any piece of work past the deadline, the course software charges you one late token. Expenditure of late tokens is governed by these rules:

- You may only use at most **two** late tokens on any given deliverable. **No submission will be accepted more than two days late.**
- Once you are out of late tokens, you may still submit assignments late with a **15 point penalty per day.** Again, submissions received after two late days will receive a zero.
- If a group deliverable is late, that will use up a late token *for each group member.*

You should think of late tokens as extensions you have been granted ahead of time and use them when you might have otherwise tried to ask for an extension.

If you experience an extraordinary difficulty, such as serious illness, family emergencies, or other extraordinary unpleasant events, your first step should be to contact your advising dean as soon as you can: explain the situation to them and ask them to contact Richard. Your dean will work with Richard to make appropriate arrangements. **IMPORTANT: The earlier you notify your dean, the more flexibility the course staff will have to make appropriate arrangements.**

Regrades and Grade Explanations

If you want to request a regrade for an objective error on our part, you must submit a private request to the course staff via Piazza explaining the error. The deadline for these requests is one week from the time the grades were posted for a given deliverable; no exceptions to this policy will be made. A regrade request may or may not result in a new grade being assigned. You are always welcome to ask for an explanation of the grade you received.

5 Collaboration and Academic Honesty

If you have **any** questions about the below policies or a specific situation dealing with academic integrity, do not hesitate to ask the instructor. All students are expected to read and adhere to the Tufts Academic Integrity Policy.

Collaboration: Projects

You will collaborate with a small team on the semester-long project; your approaches and designs may be shared with other teams, but no code, text (for project submissions), or images should be shared, partially or otherwise.

Collaboration: Homework

All homework assignments are *individual work*: while you may discuss the problems and general strategies with classmates, each student is expected to ultimately come up with their own solution. Lifting partial or complete solutions from anyone (classmates, online sources, strangers) is completely prohibited. To ensure you follow this policy, nothing should be written down, typed, or recorded in any way from your discussions with other students about homeworks. You may not work on the specific problems (i.e., produce or present problem-specific code or text) with anyone other than TAs or the instructor, and you should not use the internet in any capacity to help solve any problem. No questions, student solutions, or instructor-provided solutions should be posted online in any capacity (except as a submission to Canvas).

Academic Honesty

Any violation of the above policies will be considered cheating, and all students involved in any capacity will be forwarded directly to the Office of Student Affairs. Their sanctions range from horrible to inconceivably horrible. It's not worth it.

6 Policy on Sharing Materials

It is against Tufts policy for anyone to share any content made available in this course including course syllabi, assignments, videos, and exams with anyone outside of the course without the express permission of the instructor. Any such sharing or posting could violate copyright law or law that protects the privacy of student educational records.

7 Inclusivity, Accessibility, and Additional Help

This course is inclusive of all participants, regardless of personal identity (gender, race, sexual orientation, etc.). In the classroom and our discussion forums, everyone is expected to treat everyone else with dignity and respect. If you feel unwelcome or mistreated for any reason, please let a member of the teaching staff know so we can work to make things better.

Accommodations for Students with Disabilities

Tufts University values the diversity of our body of students, staff, and faculty and recognizes the important contribution each student makes to our unique community. Tufts is committed to providing equal access and support to all qualified students through the provision of reasonable accommodations so that each student may fully participate in the Tufts experience. If a student has a disability that requires reasonable accommodations, they should please contact the StAAR Center (formerly Student Accessibility Services) at StaarCenter@tufts.edu or 617-627-4539 to make an appointment with an accessibility representative to determine appropriate accommodations. **Please be aware that accommodations cannot be enacted retroactively; all accomodation notes must be provided to the instructor wihtin one week of the note being written to guarantee consideration.**

Academic Support at the StAAR Center

The StAAR Center (formerly the Academic Resource Center and Student Accessibility Services) offers a variety of resources to all students (both undergraduate and graduate) in the Schools of Arts and Sciences, and Engineering, the SMFA, and The Fletcher School; services are free to all enrolled students. Students may make an appointment to work on any writing-related project or assignment, attend subject tutoring in a variety of disciplines, or meet with an academic coach to hone fundamental academic skills like time management or overcoming procrastination. Students can make an appointment for any of these services by visiting tutorfinder.studentservices.tufts.edu, or by visiting students.tufts.edu/staar-center.

Religious Holy Days

We will reasonably accommodate any student who, for reasons of observing religious holy days, will be absent from class or experience any hardship in the completion of their work during the holy days. The instructor does not need to be notifed about absence from class - simply check with a classmate to get notes and any announcements. Please contact the instructor if you need any additional accommodation (such as for assignments or exams); reasonable accommodations will be allowed at no penalty.