COMP 181: Compilers

Spring 2021

https://canvas.tufts.edu/courses/26131/

Class Sessions	Tuesdays and Thursdays, 1:30pm-2:45pm EST, Tisch Library Room 302
Instructor	Richard Townsend, richard@cs.tufts.edu
Richard's Office Hours	Wed. 2:30pm-4:30pm EST, Zoom ID: richardtownsend
PhD TA	Brian LaChance, Brian.Lachance@tufts.edu
Brian's Office Hours	Fri. 2:30pm-4:30pm EST, Zoom ID: brianlachance

1 Course Overview

Summary

In COMP 181, 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 some specific compilation techniques for language-specific features. Traditional compiler optimizations may also be introduced.

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.

Logistics

This course will have in-person class sessions and Zoom office hours. Class sessions will not be recorded in any situation. If Tufts requires you to quarantine due to a positive COVID-19 test (yours or a close contact's), the instructor will be automatically notified of your absence and will give you information on how to attend and participate in class synchronously over Zoom. Due to the difficulty in teaching both in-person and Zoom students simultaneously, class sessions will only be provided over Zoom to students that cannot attend due to quarantining.

If the instructor or the university decides that in-person class sessions are no longer safe, the course will transition to a fully remote setting (synchronous Zoom lectures will replace in-person sessions).

Prerequisites

- COMP 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 understand how primitive types are represented at the machine level.

- COMP 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; you can get to the 181 page via this link or on the Syllabus page of the course Canvas site. Please post all technical questions on Piazza; post publicly if at all possible, as other students may have similar questions or be able to help you faster than the instructor. Please post privately to the instructor if your question reveals individual work 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.

We will be using the PollEverywhere software during class sessions to facilitate an active learning environment and give the instructor a sense of class-wide comprehension. After the add/drop period ends, each student will be given a PollEverywhere account to be linked with their Canvas account for the class. We will cover the use of PollEverywhere in class.

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 tutorials, 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 Module). 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 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	Feb 02	Course Overview and Intro. to Languages		
	Feb 04	Language Processors		
2	Feb 09	Essential OCaml I		
	Feb 11	Essential OCaml II		
3	Feb 16	FAKE MONDAY		
	Feb 18	Lexing I: Regular Expressions and MicroC		
	Feb 19	(Proposal due only)		Proposal
4	Feb 23	Proposal Feedback		
	Feb 25	Lexing II: Theory of OCamllex	HW 1	
5	Mar 02	Parsing I: Grammars and MicroC Lab		
	Mar 04	Project Check-in		
6	Mar 09	Parsing II: Shift-Reduce Parsing		
	Mar 11	Parsing III: Theory of OCamlyacc		Scanner & Parser
7	Mar 16	Semantic Analysis and MicroC Lab		LRM
	Mar 18	Code Generation I: IRs and LLVM	HW 2	
8	Mar 23	LRM Feedback		
	Mar 25	Code Generation II: MicroC Lab		
9	Mar 30	NO CLASS		
	Apr 01	Project Check-in	"HW 3"	
10	Apr 06	Runtime Environments I: The Stack		Hello World
	Apr 08	Runtime Environments II: The Heap		
11	Apr 13	Runtime Environments III: More Heap!		
	Apr 15	Project Check-in		
12	Apr 20	Optimization I: Intro and Examples		
	Apr 22	Optimization II: Liveness Analysis		
13	Apr 27	Wrap-Up and Opt. III: Deforestation		Extended Testsuite
	Apr 29	Project Check-in	HW 4	
14	May 04	Extra Project Office Hours		
	May 10-11	Presentations		
	May 11	Project Report Due		

The "Buffer" class listed on May 4 is 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.

The third homework has been replaced with the semantic analysis and code generation labs, grade-wise. That is, the total points received across those two labs will count as your third homework grade. The new "deadline" for HW3 on the calendar corresponds to the deadline of the latter of those two labs.

EDIT Apr 8: The content of the last four weeks have been shifted to accomodate the additional time needed to finish off the Heap concepts. The penultimate class session will now include some course wrap-up, and the final class session has been converted to extra office hours to help groups with their projects.

4 Assessments and Grading

Assessment will be based on five criteria:

1. One semester-long group project (65%): Design the goals, syntax, and semantics of a little language, and implement a compiler for that language. Graded deliverables are as follows (comprising the 65% of the entire project): Proposal (5%), Scanner & Parser (9%), LRM (7%), Hello World (5%), Extended Testsuite (9%), Project Report (30%).

2. Four Homework Assignments (15%): These written assignments will gauge your understanding of theoretical compiler concepts and further prepare you for the project.

3. Participation (10%): We will be using the PollEverywhere software to facilitate interactive learning in most class sessions. You are expected to attend class sessions to answer the questions posed. Your contributions during group check-in and feedback sessions will also be taken into consideration for this grade.

4. Peer-Assessment (10%): 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 and Project Deliverable Logistics

All homeworks 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.

Homeworks and project 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 homework assignment and project 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 free late days for any deliverable (homework or project) *except the proposal and the final project report.* It is your responsibility to track how many late days you've used over the semester. You may only use at most two late days on any given deliverable, after which you'll be given a zero for that assignment.

Each late day is exactly 24 hours: if your submission is timestamped anywhere between 11:59:01PM on the due date to 11:58:59PM the following day, that counts as the use of a late day. If a group deliverable is late, that will use up a late day *for each group member.*

After your three free late days are used up, you may still submit assignments late with a 15 point penalty per day. Again, submissions received after two late days will receive a zero.

Regrades

If you want to request a regrade for an objective error on our part, you must submit a private request to the Instructors 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: Non-Project Course Work

All homework assignments and in-class quizzes 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

In the case that we have to change to a virtual classroom setting, we may have to record class interactions. This course is designed for everyone to feel comfortable participating in discussion, asking questions, learning, and facilitating the learning of others. In order for that atmosphere to be maintained, the recordings of our conversations will only be shared with the enrolled students in the class (not posted publicly), and it is prohibited for any of us who have access to the video to share it outside the course. 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. This especially includes any posting or sharing of videos or other recordings on publicly accessible websites or forums. 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.