Overview

Welcome to CSC488H, Compilers and Interpreters. This course provides an introduction to the issues involved in implementing languages, including parsing, code generation, analyses, and optimizations. Over the course of the term, you will build a compiler for a small (or subset of a) domain specific language.

Contact Information

Instructor	Andrew Petersen
Office	DH3096
Office Hours	M 3:15-4:45 and W 9:15-10:45
Lectures	M 1-2 and 2-3
Lab	W 11-1

Textbook

The text for this course is *Engineering a Compiler* by Cooper and Torczon. I will be reading from the 2nd edition, but I expect that the first edition material is very similar.

Website and Discussion Board

The course website contains important information: the project description, lecture materials and readings, the dicussion board, and so on. (Marks, however, will be posted on Blackboard.) The website is available through Blackboard or directly, via this link: `https://mcs.utm.utoronto.ca/~peters43/488`

The discussion board should be your first stop for CSC488 information and is *required* reading. I will post important updates and announcements there, so make sure to visit the board often. In addition, each of you can post questions (and answers!) so the discussion board will be the fastest way to get help with course material or to obtain an answer to an administrative question.

To make it easier for everyone to find answers to their questions, please use good forum etiquette. Use informative titles for your posts, so that people can find relevant information. Read the posts already on the board before posting a question so that you don't post a duplicate. Finally, be professional in tone and behaviour.

Email

Please use the discussion board to ask general course-related questions. If you have a personal issue, such as needing to report an illness or discussing an alternate test date, please email me at *andrew.petersen@utoronto.ca*. Email must be sent from a U of T address; if not, you may not receive a response.

You may also use the Anonymous Feedback feature on the website to provide feedback. Since the sender cannot be determined, comments sent through the feedback system are considered public and may receive a response at the beginning of class or on the discussion board.

Accessibility

The University of Toronto and your instructors are committed to accessibility. If you require accommodations, or there is anything course-related we can do to help, please talk to us.

Marking Scheme

Work	Weight	Comment
Project: Team Signup and Proposal	-	Wed, Jan 11 @ 5 p.m.
Project Sprint 0: Language Specification	15%	Fri, Jan 27 @ 5 p.m.
Project Sprint 1: Parser	10%	Fri, Feb 10 @ 5 p.m.
Project Sprint 2: Backend	10%	Fri, Mar 3 @ 5 p.m.
Project Sprint 3: Type Extension	10%	Fri, Mar 17 @ 5 p.m.
Project Demo	15%	Wed, Mar 29 in class
Project Sprint 4: Optimization and Final Submission	20%	Fri, Mar 31 @ 5 p.m.
Term Test	20%	Wed, Mar 15 in class

The Project

For the project, you will develop a compiler for a domain-specific language of your choice. You will specify a language (or subset of an existing language) for that domain that includes (a) at least two primitive types and related operators and (b) conditional, looping, and procedure constructs. You'll also need to select a target: an output language or artifact. In the first half of the course, you will parse that language and produce output for your chosen target. In the latter half of the course, you will extend your compiler with (a) a non-primitive data structure and (b) an analysis or optimization.

The project is broken into 5 two-week sprints. At the end of each sprint, you will create a branch of your code for submission. That branch should contain a report describing your implementation and a script (or artifacts and a documented process used to obtain the artifacts) that demonstrate how your submission meets the requirements for the sprint.

In addition to the sprints, you will demo your project to other members of the departmental community on the last day of class.

The project can be completed individually or in a team of two, or in rare cases, three members. Teams are intended to remain together for the entire term. However, if there are disputes about how the workload is being shared or if a team decides they cannot continue to work together, teams may dissolve at the end of any sprint. Students will not be allowed to join existing teams, but if two students working individually wish to form a team, they may do so at the beginning of any sprint, after informing me.

Grace Days

Each sprint is due at the end of business on a Friday. However, each student has three grace days. As long as each member of the team has a grace day, the team may spend a grace day (one from each student) to extend the submission date for the sprint by one business day. That means that spending one grace day will extend the due date to Monday at 5 p.m. and spending two will extend the due date to Tuesday at 5 p.m.

Midterm and Final Exam

There is one term test, held during class on March 15. The test will cover topics from lecture and is intended to evaluate your recall of compiler terminology and your understanding of the role of the various components of a compiler and how these components interact with one another.

Academic Offenses

All of the work you submit must be completed you and your teammates. Your work must not be submitted by someone else. Plagiarism is academic fraud and is taken very seriously. The department uses software that compares programs for evidence of similar code. Please read the Rules and Regulations from the U of T Calendar (especially the Code of Behaviour on Academic Matters):

`http://www.governingcouncil.utoronto.ca/policies/behaveac.htm`

Please don't cheat. It is unpleasant for everyone involved, including us. Here are a couple of general guidelines for this class:

- You are encouraged to discuss course concepts with other students, and you may help each other with debugging issues. However, you should not share code or other artifacts that will be submitted as part of the project.
- It's expected that you will use the internet as a resource. However, you should not use code without permission (i.e., it should be licensed appropriately), and any code you use should be cited.

  To cite code, place a comment block above and below the code that was copied, even if you end up modifying it to fit into your own program. The comment block at the top should indicate the purpose of the code and its source. The comment at the bottom indicates that the end of the copied code has been reached.

  To cite the use of a tool, place a comment at the top of the file that uses the output from the tool. This comment should name the tool being used and provide attribution (the creator of the tool and a link to the source).

- The code and tools that you find online and use should only *help* you develop a solution – they should not be the solution (i.e., incorporating large sections of someone else's compiler and saying, "But it does what I want!" isn't acceptable). If in doubt, please ask me if using a particular piece of code is acceptable.