Course Title: Structure and Implementation of Modern Programming Languages (SIMPL)

Instructor(s): Dr. Siddhartha Chatterjee

TA(s): Tyler Collins, Peter Gao, Aaron Wollman.

Office Hours:
The instructor will hold virtual office hours through Zoom by appointment only.

TAs will hold virtual office hours through Zoom according to the schedule below, and by appointment.

- Tyler Collins: MF 20:00-21:00 CT.
- Peter Gao: W 15:00-17:00 CT.
- Aaron Wollman: TuTh 14:00-15:00 CT.

Course Overview/Objectives:
Programs written in high-level programming languages are the bridge between an algorithm's intent and the machine instructions realizing that intent. Driven by higher-level abstractions in modern programming languages and advances in instruction sets and micro-architectures, today's set of technologies for translating high-level program text to binary machine code have expanded significantly beyond the "classical" set of compilation technologies developed in the 1960s. This class samples the breadth of these technologies within a unified cost-benefit-risk analysis framework, and illustrates system integration by building a series of compilers for LiveOak, a pedagogical object-oriented language.

Core topics include the theory and practice of lexical analysis, syntactic analysis, code generation, language-level virtual machines, and dynamic memory management.

Prerequisites:
The student should have experience with significant programming in Java, emphasizing the use, augmentation, and integration of pre-existing components. Prior courses in programming languages, assembly language, compilers, computer architecture, and automata theory are helpful but not required.

Textbooks:
None.

List of Topics:

Week	Topics	Assignment	Exam
1	Course introduction. LiveOak, SaM.	SaM String Library	
2	Practical parsing and code generation for SaM.		
3	Theory of lexical analysis.	LiveOak-2 to SaM Compiler	
4	Theory of syntactic analysis.		
5	None.		Mid-term
6	Objects, classes, and inheritance.		
7	Implementing LiveOak-3.	LiveOak-3 to SaM Compiler	
8	Semantic analysis. Type checking.		
9	Memory management. Garbage collection.		
10	None.		Final

Assignments:
There will be three programming assignments, ranging in duration from two to four weeks. All programming assignments must be attempted and completed individually. The programming language for the assignments is Java. Assignments will be submitted on-line for grading, and will be auto-graded. Repeated submissions are allowed and encouraged. The date of the last submission will be considered for computing slack days. Plagiarism-detection software will be used to flag possible violations of academic integrity.

Additionally, several *optional* problem sets will be handed out. These are intended solely to improve your understanding of the subject matter and help you prepare for the exams. You may work on them individually or in groups. The answer key for an individual problem set will be released in the edX course two hours after the problem set is due.

Exam(s):
There will be two timed on-line exams: a mid-term and a final. Both exams will be open-book and open-notes. The mid-term exam will be held in week 5 and the final exam in week 10 of the course. Each exam will be open for a 120-hour period to allow students flexibility in taking them, and must be attempted in a single sitting. Further details will be distributed via Canvas and Piazza once they are finalized.

Grading Policy:
- Programming assignments are worth 60% of your grade.
- Exams are worth 40% of your grade (20% for mid-term, 20% for final exam).
- Exam grades may be curved. Program grades will not be curved.
- Grade cutoffs are as follows: A: 90% or higher; B: 80%-89.9%; C: 70%-79.9%; D: 60%-69.9%; F: 59.9% or lower.

- Problem sets are voluntary and do not count towards your grade. However, evidence of attempting them may be used to adjust borderline grades upwards.

Program Grade Requirements:
30 hour program
9 required hours
21 elective hours
Required courses, B- or higher.
Elective courses, C or higher.
To graduate, all students must have a graduate GPA average of at least 3.00.

Late Policy:
Five slack days will be available for the semester and may be applied (in units of full days, i.e., no fractions of days) towards programming projects. Individual projects may limit the maximum number of slack days that can be applied toward them.

Tech Requirements
- Laptop or personal computer with the following requirements
    - OS: Windows 10 or Mac 10.12/10.13/10.14
    - At least 2GB of free hard drive space
    - Dual-Core 2.4Gz Processor or better
    - 4GB RAM or better
    - Internet Connection: Cable modem/DSL or better (500kbps download, 300kbps upload)
    - Browser: Latest version of Google Chrome or Firefox (Chrome is preferred)
- Smartphone or scanner to take pictures and make PDFs of homework submissions
- Smartphone or other device capable of being used for dual-factor authentication

Academic Integrity in This Online Course
The online course format allows for multiple methods of identity verification and monitoring/detection of collusion, collaboration, and plagiarism. A violation of the course policy may include (but is not limited to) the following:
- Providing your UT EID to any other person.
- Collaborating or sharing information with another person regarding the material on any quiz, assessment, or assignment, before, during, and/or after any quiz, assessment, or assignment.
- Recording any quiz, assessment, or assignment material in any format
- Failing to properly cite language, ideas, data, or arguments that are not originally yours.
- The public (such that it can be viewed by more than one person) posting of any form of a test bank or group of questions from any assignment.
- Consulting forbidden materials or sources of information.

The University of Texas at Austin Academic Integrity principles call for students to avoid engaging in any form of academic dishonesty on behalf of yourself or another

student. Grade-related penalties are routinely assessed ("F" in the course is not uncommon), but students can also be suspended or even permanently expelled from the University for scholastic dishonesty.

If you have any questions about what constitutes academic dishonesty, please refer to the Dean of Students website or contact the instructor for this course.

You must agree to abide by the Honor Code of the University of Texas. You will not work with or collaborate with others in any way while completing any of the graded course assignments.

Accommodations
The University of Texas at Austin guarantees that students with disabilities have access to appropriate accommodations. You may request an accommodation letter from the Division of Diversity and Community Engagement, Services for Students with Disabilities.

If you have approved accommodations for the course, please contact us to arrange them. Please do this as soon as possible, so that you can have the benefit of the accommodations throughout the duration of the course.

Course Etiquette
We expect that you will treat online discussions as though you are having a civil, respectful discussion with your fellow classmates in the same classroom. Please refrain from using profanity or any euphemisms for profanity. Please do not bait other commenters or personally attack them. Please do not use sarcasm in a way that can be misinterpreted negatively. And please do not make the same point repeatedly. In short, please just respect the right of your colleagues to ask questions and discuss their opinions about the subject matter of our course on the discussion board. Violators of these discussion rules will simply be shut out from all class communications—email, Piazza, and office hours.

Academic Advisor Support
If you have additional questions or require support from an academic advisor, please contact the program coordinator at msonline@cs.utexas.edu.

ALR Spring 2021

Welcome to the home page for Automated Logical Reasoning!

Logistical Information:

Instructor:	Işıl Dillig
Time:	Tuesday, Thursday 2-3 pm
Place:	Zoom
Instructor e-mail:	isil@cs.utexas.edu
Instructor office hours:	Thursday 3-4 pm
TA:	Shankara Pailoor
TA e-mail:	spailoor@cs.utexas.edu
TA office hours:	Monday 4-5 pm
Reference books (optional):	The Calculus of Computation by Aaron Bradley and Zohar Manna; Decision Procedures: An Algorithmic Point of View by Daniel Kroening and Ofer Strichman
Course Webpage:	http://www.cs.utexas.edu/~isil/cs389L/

Course Description:

Automated logical reasoning has enabled substantial progress in many fields of computer science, including software and hardware verification, theorem proving, program analysis, and artifical intelligence. In this course, we will study widely-used logical theories and decision procedures for answering whether formulas in these theories are satisfiable. In particular, we will consider automated reasoning techniques for propositional logic, first-order logic, linear arithmetic over reals and integers, theory of uninterpreted functions, and combinations of these theories. We will also look at applications of logic in program analysis and verification.

Requirements:

- Regular class attendance is required. Please email instructor if you will not be able to attend class on a particular day.
- This course will have a combination of programming assignments and problem sets. Collaboration is **not** allowed on either.

- This offering of the course will not have exams.

Announcements:

- Our first class will meet on Tuesday, Jan 19.
- All problem sets and programming assignments will be posted on Piazza .

Syllabus:

In the Reading section of the syllabus below, COC refers to the Bradley & Manna Calculus of Computation book, while DP refers to the "Decision Procedures: An Algorithmic Point of Book" by Kroening & Strichman.

Date	Lecture topics	Notes	Reference
01/19	Introduction and basics	Lecture 1	COC 1.1-1.5
01/21	Normal forms and DPLL	Lecture 2	COC 1.6-1.7
01/26	CDCL-based SAT Solvers	Lecture 3	DP 2.2 CDCL SAT solvers
01/28	Free time for programming assignment	N/A	N/A
02/02	Practical applications of boolean satisfiability	Lecture 4	N/A
02/04	Binary decision diagrams	Lecture 5	Notes on BDDs DP 2.3
02/09	Semantics of First Order Logic	Lecture 6	COC 2.1-2.4, COC 2.7
02/11	Proof rules and properties of FOL	Lecture 7	COC 2.6
02/16	Unification	Lecture 8	
02/18	First-order theorem proving	Lecture 9	
02/23	Overview of First-Order Theories	Lecture 10	COC Chapter 3
02/25	Theory of Equality	Lecture 11	COC Chapter 9
03/02	Free time for programming assignment	N/A	N/A
03/04	Linear Arithmetic over Rationals	Lecture 12	CLRS Chapter 29
03/09	Linear Arithmetic over Integers	Lecture 13	
03/11	Nelson-Oppen	Lecture 14	COC Chapter 10
03/16	Spring break		
03/18	Spring break		
03/23	DPLL(T) Framework	Lecture 15	
03/25	Hoare Logic	Lecture 16	
03/30	Verification conditions	Lecture 17	
04/01	VCs with functions and pointers	Lecture 18	
04/06	Intro to Dafny	N/A	
04/08	Time for programming assignment		
04/13	Abstract interpretation	Lecture 19	

04/15	Guess-and-check methods	Lecture 20	Houdini Abduction
04/20	Predicate abstraction; CEGAR	Lecture 21	
04/22	Proving (non-)termination	Ledture 22	
04/27	Reasoning about concurrency	Lecture 23	
04/29	Program Synthesis I	Lecture 24	
05/04	Program Synthesis II	Lecture 25	
05/06	Wrap-up		