Catalog Description
Syntax analysis and semantic processing for a block-structured language. Lexical analysis, symbol tables, run-time management. Students will write a compiler.

Prerequisites
- Programming competence in C or C++ (e.g., via CS241) or Java (e.g., via CS351).
- CS341; comfort with machine instruction sets and assembly languages.
- CS351; comfort with the design and implementation of substantial programs.

Textbook
None required. Lectures and internet background materials will cover selected compiler topics.

Class Web Site
The public class information page is at http://www.cs.unm.edu/~ackley/554, but most class materials – announcements, lecture slides, project information, online discussion – will be in the Slack group at http://compilerconstruction.slack.com/, whose membership will be limited to currently-enrolled class members. Students should join the group promptly via an email of their choosing at http://nm8.us/s18, but note each group member's name must obviously match that on the active UNM class enrollment list, or the membership will be deleted.

Note that all students are responsible for checking the Slack group regularly for announcements and other important course-related materials!
Additionally, a separate web site will be available for turning in course work and checking grade information; details will be presented in class.

Course work
The principal course work is the semester-long project designing and implementing a compiler for a specified custom language that is mostly a subset of the C programming language. The compiler itself must be implemented in either C, C++, or Java. Project “spikes” representing steps toward the full (subset) compiler will be due at frequent intervals throughout the semester.

All projects and project spikes will be completed individually. There will be no team or group projects. Throughout the semester there will be individual code reviews during classes to assess student progress and code quality. There will be a final exam testing knowledge and mastery of the topics covered in the class.
**Academic honesty**

Software development in society at large is often a team endeavor but this class is focused on individual learning and individual achievement. For this class there are three basic guidelines:

1. **The work you turn in must be your own, in both code details and overall design.** Superficial code alterations like renaming variables or reversing if-else's do not make a design your own, though they may well demonstrate intent to deceive on your part.

2. When interacting with other students in the class: **You may help each other understand the problems, but not the solutions.** It is acceptable to discuss algorithms and data structures in general, and coding style, and the requirements of the assignment, with other members of the class, or to get a limited amount of debugging help from another.

3. When using internet and other resources outside the class: **You may only use sources that are independent of the class and the project.** For example, using a code snippet from a two-year old 'stackoverflow' question is fine, but posting questions or requests based on the project is unacceptable, because any replies would not be independent of the project.

Other examples of unacceptable behavior include: Copying another person's program with or without their knowledge, codeveloping a program, mailing all or portions of your program to another person, making your files readable so another person can copy them, reading another person's files, using another person's listing (taken from the trash, for example), having another person write any portion of your program for you.

Cheating will result in an automatic F for the entire course and turning the case over to the appropriate authorities for further disciplinary action. There will be no second chances. In cases of copying, where it is sometimes difficult to tell who was copying from whom, all students with knowledge of the cheating will be penalized except in rare circumstances. If in doubt as to whether a collaboration is cheating it is your responsibility to ask your instructor in advance.

**Class attendance**

You are expected to attend all class meetings. Beyond that, you are expected to participate in the class, for example by asking questions, raising issues for discussion, and contributing test cases. You are responsible for all material presented, including any announcements made in class or via http://compilerconstruction.slack.com/, and any handouts distributed in class.

**Grading**

The project overall will count for approximately 50% of your grade, plus 25% for the final exam, and 25% for class participation – which includes regular attendance, questions and comments, and performance during code reviews. Grades may also take into account progress over the semester. Hence, a student who improves over the semester may receive a better grade than their average might indicate.

**Turning in programs**

Project spikes will be turned in via a mechanism discussed in class, on or before 5pm MT on the assigned due date. Late turn-ins will be accepted for a period of time, but they will be marked down 15% per 24 hour period or fraction thereof that the turn-in is late, including weekends and holidays. For example, consider a project spike that is due on a Friday. If the turn-in occurs at
6:17pm on Saturday, it is two days late and would thus have a maximum possible score of 70%.

However, you are allowed three (3) free "late days" for the entire semester. This means that you can, for example, hand one spike in three days late without loss of grade, or one spike two days late and another spike one day late, and so forth. Any of us may have a family crisis, scheduling conflict, unanticipated travel, or illness during the semester; the late days are intended to cover such emergencies. **You do not explicitly request to use your late days:** At the conclusion of the class, your three free late days will automatically be assigned to cover your actual late turn-in days, if any, in a manner which is most advantageous to your overall course grade. Problems such as serious illnesses that require a longer delay will be dealt with on a case by case basis and must be documented. In addition, makeup of a missed examination will only be possible where the student has documented a serious and unanticipated problem.

**Design, style, productivity**

In many ways, successfully building a compiler, like any large edifice, is as much about focusing on effective construction practices as it is about focusing on the artifact itself. As the spikes progress, you will need to modify and extend your code in radical ways; if the code or the design is sloppy, you will probably be unable to keep up. For success and for grading, everything counts, from choosing and maintaining accurate names for classes, methods, and variables, to maintaining good design and modularity, to using productive work practices and maintaining a steady pace.

Clean design and code, and good tests, are critical. Finding bugs is riskier than avoiding them.

--- CUT HERE --- CUT HERE ---

Please complete, sign, and return this portion of the page to Professor Ackley (KEEP above and previous page for reference.) Your turn-ins will not be graded without this signed agreement.

Your full name (print legibly):

____________________________________________________

Your UNM student ID number:

____________________________________________________

What name do you prefer people call you:

____________________________________________________

A PIN for you to access your C454/554 grade information (4-8 decimal digits. Don't reuse an 'important' PIN! Print large!):

____________________________________________________

**Student Declaration:** I have read and understood this CS454/554 S18 Statement of Policy and I agree to abide by its contents.

Signed:____________________________________________  Date:______________

http://www.cs.unm.edu/~ackley/554