In about 500 words, describe your independent study and how you intend to complete it. Include your motivation for pursuing the project and why it cannot be done within the framework of a regularly offered course. Describe in some detail what you intend to produce over the course of the independent study, e.g., a paper(s), a performance(s), etc. Provide an estimate of your weekly working hours and weekly contact hours with your faculty supervisor during the course of the semester project.

Spreadsheets are an important application used widely by end-users. Spreadsheets are a fast and intuitive way to record and analyze data. Spreadsheets are an easy-to-use but very powerful tool for computer programming. At the same time, spreadsheets lack some of the rich tooling available for other programming languages like Python and C++. In this independent study, I seek to close that gap by creating a tool to address the widespread problem of data transformation in spreadsheets.

Data transformation tasks encompass any kind of rearrangement of the spreadsheet into a format that is more easily analyzed. Data transformation tasks are difficult-to-impossible in most spreadsheet environments. Spreadsheet users often end up performing the tasks by hand, rewriting the spreadsheet and enduring the tedium of copying and pasting data from one place to another. This task is readily suited to computer automation. This issue of data transformation was explored by Professor Dan Barowy in his 2014 research paper *FlashRelate: Extracting Relational Data from Semi-Structured Spreadsheets Using Examples*. The paper introduced the Flare programming language, which is useful for simple spreadsheet data transformations.

Currently, only a slow, proof-of-concept interpreter for Flare exists. I propose to explore the theory and implementation for a high-performance compiled version of Flare instead. By exploring and implementing a compiler for Flare, I will learn how both Flare and compilers work from the inside out. Looking forward, this experience will provide me with a long-lasting transferable knowledge of compilers and a deep familiarity with the implementation of programming languages. Along the way, I will learn the OCaml programming language, developing functional programming skills that are not often emphasized in traditional programming courses.

Furthermore, implementing the Flare compiler will allow me to gain experience designing software. The development of a Flare compiler is a greenfield project: There is no publicly available reference implementation; there is no precedent to constrain the work. This grants me the opportunity to exercise creativity by designing software myself, crafting my own timeline, and dealing with unforeseen problems. This independent study uniquely provides a chance to practice software design skills, to exercise creative expression within software design, and to develop a semester-long self-directed project oriented toward end-users. These are fundamental parts of the independent study which are not present in regularly offered courses.

Estimated weekly working hours: 10 hours
Estimated weekly contact hours: 1 to 2 hours

Provide a week-by-week plan for your project, a reading/research list and, if relevant, a description of the methodology and sources you will use for your work. If the project includes being away from campus for any part of the semester, explain fully.

Reading and Research:

Dan Barowy et al., *FlashRelate: Extracting Relational Data from Semi-Structured Spreadsheets Using Examples*

This research inspired the independent study.

Andrew W. Appel, *Modern Compiler Implementation in ML*

This is a textbook on compilers.

Michael Sipser, *Introduction to the Theory of Computation, 3rd Edition*

This textbook explains the theory of regular expressions.

Russ Cox, *Regular Expression Matching Can Be Simple and Fast*

This is an article on high-performance implementations of regular expression matching.

LLVM Project, llvm.org

This is the official website of the LLVM project, which is an important unit of software in my project. I plan to use this as a reference as I work with LLVM

Adrian Sampson, *LLVM for Grad Students*

This is an article that provides a quick introduction to LLVM and how to get started using it.

Wilfred Hughes, *My First LLVM Compiler*

A brief article on creating a compiler for a very simple programming language using LLVM.

Arpan Sen, *Create a working compiler with the LLVM framework*

An introduction to using the LLVM API to generate LLVM IR.

Timeline:

First third:

Get familiar programming in OCaml

Implement regular expression matching algorithms in OCaml using an interpreted approach

Implement the approaches described in the Russ Cox article

Research and select a unit testing framework for the in-development compiler

Implement Flare matching algorithms in OCaml using an interpreted approach

Learn how to use parsers and parser generators

Second third:

Learn about LLVM and the associated intermediate representation

Study the following compiler phases in the Appel textbook, implementing and testing each phase as I go

- Abstract syntax
- Semantic actions
- Intermediate representations
- Instruction selection
- Register allocation

It may be useful to start by implementing these phases for regular expressions since they are a simpler version of Flare

Final third:

Register allocation

Instruction selection

Other advanced topics in compilation