SI 413: Programming Languages and Implementation

Course Policy, Fall AY2026

Instructors:

- Prof. Daniel S. Roche, 447 Hopper Hall, x36814, roche@usna.edu (Coordinator).
- LCDR Christopher Norine, USN, 442 Hopper Hall, x36826, norine@usna.edu.

Course Description: This course examines basic concepts underlying the design of modern programming languages: types, control structures, abstraction mechanisms, inheritance, concurrency and constructs for programming. The course includes programming assignments in several languages.

Credits: 2-2-3

Pre-requisites: SI342 and (IC312 or SY301)

Learning Objectives:
Upon completing this course, students should be able to:

1. Understand the functional programming paradigm and be able to solve problems in a functional language (supports outcome CS-6).
2. Develop a vocabulary for describing programming languages (supports outcome CS-6).
3. Understand lexical analysis, parsing, and basic interpretation (supports outcome CS-6).
4. Implement a simple interpreter (supports outcome CS-6).
5. Learn a new programming language independently and use it to solve basic tasks (supports outcome CS-6).

Student Outcomes:
Graduates of the program will have an ability to:

1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.

2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

3. Communicate effectively in a variety of professional contexts.

4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.

5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.

CS-6. Apply computer science theory and software development fundamentals to produce computing-based solutions.

DS-6. Apply theory, techniques and tools throughout the data analysis lifecycle and employ the resulting knowledge to satisfy stakeholders' needs.

Course schedule:

The course is structured around a single task: designing and implementing a new programming language. Each unit is centered on a few additional programming language capabilities and a series of four tasks:

(1) Designing and specifying the syntax and semantics of a novel language to support the required capabilities;
(2) Writing example programs in the novel language (which will later be used for testing);
(3) Implementing a REPL-based interpreter for the language in Java;
(4) Implementing a compiler for the language using LLVM.

To support these concrete tasks within each unit, we will also introduce various new concepts and examples from existing programming languages.

- **Unit 1: Just Strings** (3 weeks)
  - Capabilities: literals, input/output, unary and binary operators.
  - Concepts: Specifications, undefined behavior, syntax and semantics, expressions and statements, interpreters and compilers
  - Languages: LLVM IR
- **Unit 2: Variable and Booleans** (3 weeks)
  - Capabilities: Boolean literals, string-boolean and logical operators, named variables
  - Concepts: Scanning and Parsing, symbol table, memory allocation, errors, type safety
  - Languages: Scheme
- **Unit 3: Taking Control** (2 weeks)
  - Capabilities: Conditionals, loops
  - Concepts: Abstract syntax trees, basic blocks, control flow graph, single static assignment, phi nodes
  - Languages: Rust
- **Midterm exam** (Wednesday, October 15)
- **Unit 4: Functions** (3 weeks)
  - Capabilities: Function calls, lexical scope
  - Concepts: Heap allocation, anonymous functions, frames and closures
  - Languages: Scheme
- **Unit 5: Numbers** (2 weeks)
  - Capabilities: Integer literals and operations, builtin functions, type checking
  - Concepts: Static and dynamic typing
  - Languages: Haskell
- **Unit 6: OOPs** (2 weeks)
  - Capabilities: Objects with basic inheritance
  - Concepts: Named types, dispatch, polymorphism
  - Languages: Esoteric languages (Piet, Befunge, iogii)

Updates to the course policy: In case this course policy needs to be changed during the semester, students will be notified by email and verbally during class. The current version will always be posted on the course website.

Textbooks:

- Michael L. Scott. *Programming Language Pragmatics*, Morgan Kaufmann Publishers. Optional but recommended. The current edition is the 4th (2015), but any edition is OK. Page numbers in the notes are from the 3rd edition.
- Abelson, Sussman, and Sussman. *Structure and Interpretation of Computer Programs*, MIT Press. Out of print but available used or for free online.
- Aho, Lam, Sethi, and Ullman. *Compilers: Principles, Techniques, and Tools*, Pearson, 2006. Completely optional, recommended for those wishing to gain deeper knowledge of compilers.

Course Website: The primary website for course content is
https://faculty.cs.usna.edu/~roche/413/
This site is only available on the Yard, on the USNA intranet. To access exactly the same content from anywhere else, you can also use the following unofficial mirror:
https://www.roche.work/413/

Extra Instruction: Extra instruction (EI) is strongly encouraged and should be scheduled by following the instructions on your instructor's homepage. EI is not a substitute lecture; students should come prepared with specific questions or problems.

Collaboration: The guidance in the Honor Concept of the Brigade of Midshipmen and the Computer Science Department Honor Policy must be followed at all times. See https://www.usna.edu/CS/resources/honor.php. Specific instructions for this course:

- Homework: The purpose of homework assignments is to provide example problems of the type that will appear on written exams. Solutions will be provided and usually reviewed in class, and credit will be given according to completion only. Students are strongly encouraged to honestly try solving problems themselves before using any assistance from other students, web searches, or AI tools.
- Labs: These represent the main work of the class, and each student is required to completely understand the work they turn in. Students should expect to be asked to explain in detail how and why their code works to demonstrate this understanding.

  Collaboration and assistance from other humans is permitted but should be limited to conceptual discussions and debugging help, never sharing or copying even small parts of working code. Any assistance received must be clearly and specifically documented.
- Exams: No collaboration is allowed. Each student may prepare and bring a single study sheet to the midterm and final exams, but these must be prepared individually (no photocopies). Any group study guides should be shared with the instructor.

All collaboration and outside sources should always be cited. The same rules apply for giving and receiving assistance. If you are unsure whether a certain kind of assistance or collaboration is permitted, you should assume it is not, work individually, and seek clarification from your instructor.

Use of Generative AI: The use of generative AI tools for homework is permitted but discouraged before honestly trying to solve the problems by yourself. (Remember the purpose of homework is to help you prepare for exams.)

For labs, use of the USNA-provided generative AI tool Gemini is permitted with the following limitations:

- Complete transcripts of any Gemini conversation relating to your work on this lab must be turned in alongside the assignment. To get a transcript in markdown form, just ask Gemini to "give me a complete transcript of this entire conversation in markdown, including all prompts and responses verbatim."
- Do not ask Gemini to complete your entire assignment, but only for help with specific small parts or understanding of concepts. Generally only small pieces of code (like one or two lines) should be copied directly into your work, and as stated above you will be expected to explain (without using Gemini) how this code works after turning it in.

Classroom Conduct:

Everyone in the classroom will show appropriate respect to each other at all times.

The section leader is responsible for recording attendance, bringing the class to attention, notifying the CS department office if the instructor is more than 5 minutes late, and directing the class in useful work in the instructor's absence.

Drinks are permitted, but they must be in closable containers. Food, alcohol, and tobacco (of all kinds) are prohibited. The use of laptops, tablets, or other devices during class is at the discretion of the instructor; such use must be related to the class and should never serve as a distraction to other students.

The class will follow the non-attribution of communication practice (Chatham House Rule).

Absences:

Students are responsible for all class material. Notes will be posted for each lecture, along with recommended readings. However, this material is not exhaustive and students missing class should arrange to copy notes from a classmate.

Late Policy: No work will be accepted for a grade after stated deadlines. The same deadlines apply to all students, even in cases of excused absences. Students travelling may need to work ahead in order to get their work submitted on time.

Exceptions to this rule are possible under exceptional circumstances, at the discretion of the instructor, and especially for unforeseen health or family issues. Just ask.

Grading:

For labs we will use a form of *specifications grading*. All parts of the labs (language specification, example programs, interpreter implementation, and compiler implementation) will have specific standards which must all be met in order to earn a stated number of points for that part. Until the standard is met, there will be no partial credit. However, provided sufficient effort and progress are demonstrated and subject to the end of classes date, any incomplete parts may be revised and resubmitted within one week (and re-revised, under the same conditions).

Your final grade will be computed as follows:

- 5%: Weekly homeworks (completion-based grading)
- 50%: Labs (language specification and implementation)
- 10%: Midterm exam
- 35%: Final exam

6 and 12 week grades will be computed based on this same breakdown, scaled according to the amount of work completed so far.

Plus/minus grades will be assigned based on the following numerical cutoffs:

	-		+
A	90–92	93–100	
B	80–82	83–86	87–89
C	70–72	73–76	77–79
D		60–66	67–69
F		0–59	

Submitted: Prof. Daniel S. Roche