

# CSCI-400: Principles of Programming Languages (PL)

**Semester Year:** Spring 2025

**Class Time:** 12:30-1:45pm Tue/Thu

**Class Location:** Green Center Metals Hall

**Instructor:** Dr. Aysu Betin Can

**Office Location:** CTLM 246D

**TAs:** Blake Aduato, Alexander “Alex” Chenot, PJ Wetherel, Ronan Rael

## 1 Overview and Outcomes

### 1.1 Course Description

In this course, we will examine **fundamental concepts** and techniques behind programming languages.

**This is a challenging but very exciting course! We will be learning what happens under the hood of language abstractions. After all, a programming language is the most fundamental tool for computer scientists and software engineers.**

This course will improve your skills as a programmer and will deepen your understanding of how programming languages are designed and implemented. Learning these common and fundamental abstractions is valuable as it will make it easier to learn and use different languages, some are not invented yet.

We will illustrate these concepts within the realm of **functional programming**. We will use OCaml as our language of choice and implement the examples using OCaml.

In the lab assignments, you will submit OCaml programs implementing the concepts we have discussed in the lectures. After the midterm, you will implement a JavaScript interpreter in OCaml. Rather than starting from scratch, you will complete a partially developed interpreter.

This course will emphasize *functional programming* for several reasons:

- Functional programming generalizes many programming constructs you have previously studied;
- Functional programming predicts the future of language development. Many programming languages that began in another style (imperative, object-oriented, etc.) are gradually gaining functional programming features over subsequent revisions;
- And functional programming helps us write correct (bug-free) programs. Functional languages offer many tools to reason about and ensure program correctness, both informally (in our heads) and formally (with an algorithm).

## 1.2 Learning Outcomes

Through the activities in this course, you will learn the following:

**Part I** Implement programs in the *functional* style and in a functional language, using functional programming features to support program correctness, including:

- understand the evaluation of functional programs, as based on the *lambda calculus*;
- write recursive functions, including recursive handling of structures such as lists and tree;
- use *static typing* and *pattern matching* syntax;
- use common higher-order functions—such as `map`, `filter`, and `fold`—write new higher order functions, and use *lexical closures*;
- use and implement *algebraic data types*;
- use, implement, and analyze *persistent data structures*.

**Part II** Implement the core of a programming language, including:

- use a lexer and parser generator to create a lexer and parser;
- implement expression evaluation;
- implement variables;
- implement function definition, function application, and lexical closures.

**Throughout** Analyze and evaluate questions about programs and programming languages:

- explain programming language theory terms including lexical closure, lexing, parsing, big-step semantics, small-step semantics, lexical environment, type checking, type environment, type inference, and references;
- relate common programming constructs and idioms to the *lambda calculus*, including variable definition, lexical closures, and currying;
- reason inductively about recursive functions and data structures;
- prove amortized running times for a data structure or algorithm;
- analyze, compare, and contrast the suitability of mutable/ephemeral/imperative vs. immutable/persistent/functional programming approaches and data structures for new problems.

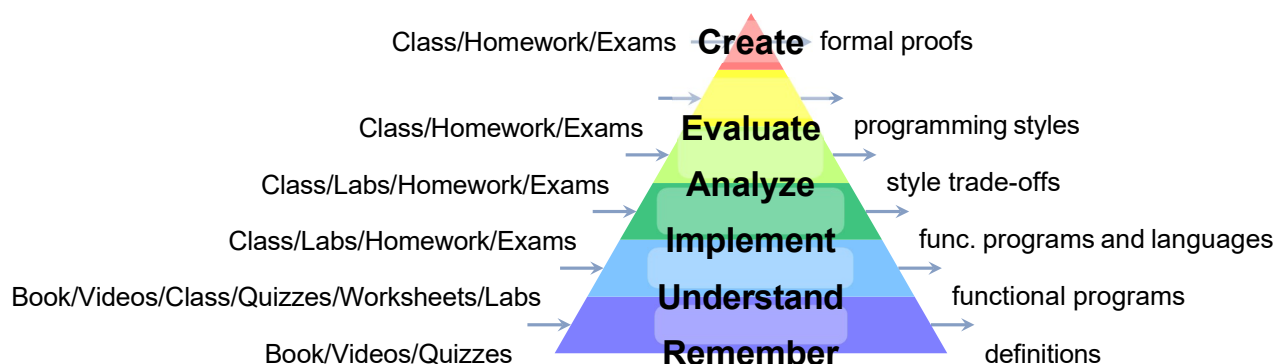


Figure 1: Bloom's Taxonomy of Learning Activities and Outcomes

## 2 General Course Information

**Prerequisites** The prerequisites are CSCI-358: Discrete Math and CSCI-306: Software Engineering. If you have somehow registered without completing the prerequisites, please drop the course.

**Textbooks and References** *Studying the textbook(s) is a valuable (and expected) learning activity.*

- **Primary Textbook** (main reference for the course) Clarkson et al. [OCaml Programming: Correct + Efficient + Beautiful](#).
- **Supplemental Textbooks** (references for some advanced topics)
  - Chris Okasaki. [Purely Functional Data Structures](#). Cambridge University Press. 1998. ISBN-13: 978-0521663502.
  - Benjamin Pierce. [Types and Programming Languages](#). 2002. ISBN-13: 978-0262162098
  - Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. *Compilers: Principles, Techniques & Tools*. 2006. ISBN-13: 978-0321486813.
- **Programming References**
  - [OCaml Tutorial](#)
  - [OCaml Manual](#)
  - [JavaScript Specification](#)
  - [Mozilla Developer Network](#)

**Office Hours** See CANVAS course homepage.

### Online Resources

- [Canvas](#): Written assignment submissions, Grades, Announcements
- [Ed Discussion](#): Questions, Discussion, Homework/Project help
- [Course Github Organization](#): Project code distribution and submission
- [Isengard](#): ITS-managed Linux server with shell access
- Zoom: Remote office hours

**Technology Requirements** This course assumes you are able to access a GNU/Linux system (e.g., Debian, Ubuntu). If you do not run Linux on your personal workstation, you may use the ITS-managed [Isengard](#) server or install a virtual machine. For technical support please ask the TAs. You will need to bring a device to class for coding and (electronic) discussion.

### Who should I email/contact?

- **Miscellaneous basic policy questions:** (when is the midterm? when is an assignment due?): Re-read the syllabus, check Canvas assignments, check Ed Discussion announcements, and ask any additional questions on Ed Discussion.

- **Help with assignments or course topics:** Ed Discussion, TA office hours, or instructor office hours. Private post on Ed Discussion if the matter should be hidden from other students (e.g., something about your code or questions about your grade)
- **Solutions to worksheets:** Slides with completed worksheet exercises will be posted as activity slides (for example L01-basics-activity-postlecture.pdf) after the lecture.
- **Anything sensitive or confidential:** (e.g., a health issue) Email the instructor about the issue and/or to schedule a meeting to discuss the issue.
- **Concerns/suggestions about course procedures:** Email the instructor or TAs about the issue and/or to schedule a meeting to discuss the issue.

**Acknowledgments** I would like to express my sincere gratitude to Prof. Dantam for generously allowing me to use these lecture materials, including the syllabus, for this course.

This course draws heavily from Prof. Jed McClurg's offerings in Fall '19, '20, and '21. The current version of the course is inspired by Prof. Bor-Yuh Evan Chang's Programming Languages course and Cornell's course and associated textbook CS 3110: Data Structures and Functional Programming, which itself is rooted in the legendary Structure and Interpretation of Computer Programs.

### 3 Assessments and Grading

The course score (percentage) will be computed as a weighted average of scores (points received over points possible) as follows:

Participation	10%	( $w$ )
Quizzes	10%	( $q$ )
Homework	10%	( $h$ )
Labs	35%	( $\ell$ )
Midterm Exam	15%	( $m$ )
Final Exam	20%	( $f$ )
Total	100%	

$$\text{score} = 0.1 \left( \frac{w_{\text{recv.}}}{w_{\text{poss.}}} \right) + 0.1 \left( \frac{q_{\text{recv.}}}{q_{\text{poss.}}} \right) + 0.1 \left( \frac{h_{\text{recv.}}}{h_{\text{poss.}}} \right) + 0.35 \left( \frac{\ell_{\text{recv.}}}{\ell_{\text{poss.}}} \right) + 0.15 \left( \frac{m_{\text{recv.}}}{m_{\text{poss.}}} \right) + 0.2 \left( \frac{f_{\text{recv.}}}{f_{\text{poss.}}} \right)$$

**Worksheets and Participation** Most lectures will have a worksheet to practice the material. Scan or photograph the worksheet (or complete it electronically) and submit it on Canvas. Your participation grade will be based on making an honest effort on the exercises.

**Quizzes** Most lectures will include a quiz, graded on correctness.

**Exams** There will be a midterm exam around the middle of the semester and a cumulative exam during finals week. The exams are an evaluation tool whose purpose is to produce a distribution of scores across all students that best distinguishes the extent of learning. Thus, please expect the exams to be very challenging.

**Homework** There will be 6 homework assignments to practice mathematical topics covered in the course.

**Labs** There will be 8 programming projects (labs) to practice functional programming use and implementation.

**Letter Grades** Course letter grades will be based on a curve. It is expected—but not guaranteed—that score distributions will be normally distributed and letter grades will correspond to university and department norms.

**Additional Letter Grade Criteria** In addition to the scoring and letter grade determination discussed above, to pass the course **all of the** following criteria should be met:

1. **midterm exam score must be >35%**
2. **final exam score must be >35%**
3. **weighted average of final and midterm exam scores must be >45%**
4. **your overall course score must exceed the passing threshold.**

For example: midterm 40%, final 50% and total score is C (usually above 72) passes; midterm 50%, final 40% fails ; midterm 70%, final 34% and total score 80 fails.

Grade distribution of the last term: 41% As, 35% Bs, 17%Cs, 5%Ds  
Mean of Course Total was 82.5

**Late Policy** Late work will not be accepted. However, submissions made by the end of the same day (until 11:59 PM) will be accepted with a 10% penalty.  
Please start projects and assignments early.

**Fairness** It is important to evaluate all students as evenly as possible. While we will attempt to accommodate disabilities and extenuating circumstances (physical/mental health, school-related travel, job requirements of self-supporting students, etc.) to the greatest possible extent, it would be unfair to offer any further special treatment.

**Grading Corrections** Grading changes will only be made for grading errors. It is not possible to change grades in response to disagreements about point allocation, partial credit, letter grade cutoffs, etc., because such changes would be unfair to the rest of the class. Grading corrections will only be made for the following errors:

1. *Arithmetic*: The grader incorrectly summed your points.
2. *Code*: An error in the grading environment or scripts incorrectly tested your code.
3. *Written*: The grader incorrectly understood your answer.

The deadline to request a grade correction is two weeks from the date an assignment is returned or the grade is posted.

### Projects Expectations and Grading

- Code must produce the correct output to receive credit. Incorrect output, no output, compilation errors, or runtime errors will not receive credit. **Please double-check your submitted code to ensure that minor errors will not result in major test failures.**
- Code tests will include edge cases. Think through all possible conditions for your program.
- Projects may include a peer evaluation of group members. In cases where there is evidence of

unequal contributions among group members—based on peer evaluations, git commit history, student interviews, etc.—the instructor may adjust project scores to better reflect individual contributions.

































**Written Work** Format and submit your written work as follows. Improper submission or formatting may result in a penalty on assignments.

- For FERPA compliance, all work submitted on physical paper must include a cover sheet that contains only your name and no answers or other work. Electronic submissions do not need a cover sheet.
- Write your name on *every page* of all written work. If the work cannot be matched to you, you cannot receive credit for it.
- Handwritten work must be *clearly legible* to receive credit.
- Submit electronic homework, reports, etc. in PDF format. Do not submit word processor files because these are inconsistently formatted by different software.
- Work must be readable when printed in black and white.

## 4 Tentative Schedule

Sure to change as the semester progresses.

### 4.1 Tentative Lecture Schedule

W	Date	Topic	Files	References
Part I: Functional Programming				
1	Jan 7	00: Programming Languages Introduction		Clarkson 1
	Jan 9	01: Basics of Programming Languages	 	Clarkson 2, Pierce 5.1
2	Jan 14	02: Programming with Lambda	 	Pierce 5.2
	Jan 16	03: Linux and Git (Lab 0 Day)		Git Docs
3	Jan 21	04: Functional Programming	 	Okasaki 1–2
	Jan 23	FP cont'd + 05: OCaml and Javascript (Lab 1 Day)		OCaml tutorial & ref., JS spec, MDN
4	Jan 28	Career Day (no class)		
	Jan 30	06: Higher-Order Functions	 	Clarkson 4
5	Feb 4	Lab 2 Day – HoF		
	Feb 6	07: Algebraic Data Types	 	Clarkson 3, Pierce 11
6	Feb 11	ADT cont'd +08: Persistent Data Structures	 	Clarkson 5.6, 8.3, Okasaki 2–3
	Feb 13	Persistent DS+ Lab 3 Day		
7	Feb 18	President's Day (no class)		
	Feb 20	09: Lazy Evaluation		Clarkson 8.4, 8.5, Okasaki 4
8	Feb 25	Lab 4 +10: Mutability	 	Clarkson 7, Pierce 11.2–11.3, 13
	Feb 27	Q&A for midterm		
9	Mar 4	Midterm Exam		
Part II.a: Language Implementation—Syntax				
	Mar 6	12: Syntax	 	Clarkson 9.2, Aho 4
10	Mar 11	13: Lexing	 	Aho 3
	Mar 13	Lexing continued		
11	Mar 18	Spring Break (no class)		
	Mar 20	Spring Break (no class)		
12	Mar 25	14: Parsing	 	Aho 4.4, 4.5, 4.8
	Mar 27	Lab 5 Day -ocamlyacc		
Part II.b: Language Implementation—Semantics				
13	Apr 1	17: Semantics	 	Clarkson 9.3, Pierce 3.4–3.5
	Apr 3	16: Induction +Lab 6 Day	 	Clarkson 6.7, 6.8, Pierce 2.4, 3.1–3.3
14	Apr 8	18: Environments and Scope	 	Clarkson 9.4, Pierce 3.4–3.5, 7
	Apr 10	Lab 7 Day		
15	Apr 15	19: Typing	 	Clarkson 9.5, Pierce 8–9, 22
	Apr 17	20: Type Inference	 	Clarkson 9.6, Pierce 8–9, 22
16	Apr 22	Lab 8 Day		
	Apr 24	Garbage Collection		
17	Apr 29	Final Review Q&A		
18	May 7	Finals Week		
	May 9	Finals Week		



## 4.2 Tentative Lab List

- Lab 0: Git and OCaml “Hello world”
- Lab 1: Functional programming basics
- Lab 2: Higher order functions
- Lab 3: Algebraic data types and Persistent data structures
- Lab 4: (tentative) Amortized data structures
- Lab 5: Lexing and Parsing
- Lab 6: Semantics and Evaluation
- Lab 7: Environments, Functions, and Lexical Closures
- Lab 8: References and state mutation

## 4.3 Tentative Homework List

- HW 1: Lambda Calculus
- HW 2: Types and Higher-order functions
- HW 3: Persistent data structures
- HW 4: Syntax
- HW 5: Induction
- HW 6: Type Inference

# 5 Policies

## 5.1 Mines Policies and Resources

See [mines-policies-and-resources.pdf](#) in Canvas

## 5.2 Generative AI Policy

Generative Artificial Intelligence (genAI) tools such as ChatGPT are important resources in many fields and industries. Because these tools will be used in professional and personal contexts, we believe it is valuable for you to engage critically with these tools and explore their use in generating content submitted for evaluation in this course, including worksheets, homework assignments, and lab projects. You remain responsible for all content you submit for evaluation. You may use genAI tools to help generate ideas and brainstorm. However, you should note that the material generated by these tools may be inaccurate, incomplete, biased, or otherwise problematic. We encourage you to consider how genAI complements, supplants, or fails to replace your contributions and abilities.

If you include content (e.g., ideas, text, code, images) that was generated, in whole or in part, by Generative Artificial Intelligence tools (including, but not limited to, ChatGPT and other large language models) in work submitted for evaluation in this course, [you must document and credit](#)

[your source](#). Failure to properly cite sources, including AI tools for generating content, would be considered Academic Misconduct in violation of [Mines Academic Integrity/Misconduct Policy](#).

Additional language regarding the potential pitfalls of genAI, strategies for citation, and methods for centering the human dimension of learning can be found in [this resource](#) provided by the Trefny Center.

### 5.3 CS Collaboration Policies

See CollaborationPolicy.pdf in Canvas

### 5.4 Course Policies

#### 5.4.1 Communication

Canvas Announcement are frequently used about course logistics. Ed Discussion is the primary communication tool used for questions and discussion. Students are expected to regularly monitor Ed Discussion, Canvas Announcements, and their university email (at least once a day) for announcements and changes such as modifications to class meetings. The instructor will attempt to give at least 24 hours notice (and more if possible) for such changes, but emergency situations may not allow such advance notice. If in doubt, check your email and Ed Discussion.

#### 5.4.2 Laptop and Smartphone Policy

- Lecture slides are posted in advance. You are strongly encouraged to use your laptop or phone to follow along during lecture and to review slides during exercises.
- Note-taking on laptops, tablets, etc. is welcome if you find it useful.
- Please refrain from using laptops, phones, etc. for non-class activities, e.g., email, web browsing, games, during classtime, as it is distracting to other students.
- Some class activities (e.g., coding activities, lab days) require the use of a laptop.

#### 5.4.3 Netiquette

##### Text DOs

- Ask questions and engage in conversations as often as possible—feel free to contact the instructor and TAs via the discussion forum for questions.
- When asking “tech support” questions, provide sufficient detail to diagnose and, if possible, reproduce the issue, including commands that were run, output of those commands, log files, and operating system and software versions.
- Be patient and respectful of others and their ideas and opinions they post online.
- Remember to be thoughtful and use professional language. Keep in mind that things often come across differently in written text, so review your writing before posting.
- Be prepared for some delays in response time, as “virtual” communication tends to be slower than “face-to-face” communication. Ask questions well in advance to deadlines to ensure sufficient time for a response.
- If the instructor does not respond to an important email for a few days, please send a **reminder**.

Faculty receive a large number of emails, and sometimes messages get lost or overlooked.

- Contact the instructor if you feel that inappropriate content or behavior has occurred as part of the course.

### **Text DON'Ts**

- Use inappropriate language—this includes, but is not limited to, the use of curse words, swearing, or language that is derogatory.
- Post inappropriate materials—for example, accidentally posting/showing a picture that is not appropriate for the course content.
- Post screenshots (images) of text output. Instead, post text as text. Compared to text, screenshots are slower to download, harder to read, and cannot be copy/pasted.
- Post in ALL CAPS, as this is perceived as shouting, and avoid abbreviations and informal language (e.g., “I’ll C U L8R”).
- Vent, rant, or send heated messages, even if you feel frustrated or provoked. Please instead communicate any specific concerns privately to the instructor or TAs; we want to improve the course and to accommodate any extenuating circumstances. Similarly, if you should happen to receive a heated message, do not respond to it.
- Except for course content questions on Ed Discussion, send an email or post to the entire class, unless you feel that everyone must read it.

### **Video DON'Ts**

- Post zoom links publicly, on social media, etc. Bad actors may join the meeting and post distracting or inappropriate material.
- Post off topic messages in the chat. It is distracting to others.
- Share private windows such as personal email.

## **5.4.4 Privacy and FERPA**

The university and instructor value students’ rights to privacy, and this course must specifically comply with the Family Educational Rights and Privacy Act (FERPA). To support FERPA compliance, please mind the following:

- DO NOT submit any assignment as physical paper. If there is such as need, include a cover sheet on all work submitted on physical paper. The cover sheet must have the student’s name and no answers or other work. Electronic submissions do not need a cover sheet.
- Use Canvas for electronic communication containing specifics about grades. Canvas is the system chosen by the university to manage students’ grades.
- Do not disclose the private information (e.g. grades) of other students.

### 5.4.5 Collaboration Policy

- Worksheets may be completed in groups. You are encouraged to discuss worksheet exercises with others in the class.
- Homework assignments must be an individual effort. You may not copy or share solutions. However, per the CS collaboration policy above, you may consult others in the class under the “empty hands” requirement.
- Projects may be completed with your project group. Per the CS collaboration policy above, you may consult other groups in the class under the “empty hands” requirement. Copying code will be considered academic misconduct. **All members of the group must know the project code.** There will be exam questions based on the project. I encourage everyone to do every part of the code. Previous semester students only focusing their own part of code had difficulties in the exam.
- Exams must be an individual effort. Copying solutions or consulting others on an exam will be considered academic misconduct.

## 6 FAQ

- **Q:** Is the textbook “required?”  
**A:** Studying the textbook(s) is an expected learning activity for this course.
- **Q:** What’s on the exam?  
**A:** Exam questions will be similar to homework assignments and activity slides. Exams will focus on evaluating understanding, application, and synthesis of the course topics (i.e., the upper levels of [Bloom’s taxonomy](#)). Questions will not focus on memorization, but one must know the key definitions and concepts to apply them. For the midterm, all topics covered up to the exam may be included. The final will be cumulative but will focus on topics covered after the midterm. The instructor will post a specific list of topics after preparing each exam, typically about a week before the exam date.
- **Q:** When is the midterm?  
**A:** Please see the tentative schedule in this document for an approximate time.
- **Q:** When/where is the final exam?  
**A:** The registrar schedules all final exams. Please see the [registrar’s website](#).
- **Q:** The syllabus says that Linux is the supported platform for course projects, but can I use macOS?  
**A:** MacOS does provide some unix-like features, but it also does some things differently. If you encounter a mac-specific problem, the instructor and TAs won’t be able help. In other words, it might work just fine, but if it doesn’t, you’ll have to figure it out yourself.

- **Q:** Which Linux distribution and version is best?  
**A:** Debian and Ubuntu (which is based on Debian) are good choices for distributions and have easy and robust package managers. Debian “Stable” and Ubuntu “Long-term Support (LTS)” versions will have the most thorough testing and least likelihood of encountering problems.
- **Q:** Debug my code for me.  
**A:** The instructor and TAs are here to help you with projects but typically cannot do the job of debugging for you. Plus, learning how to debug your own code is an absolutely necessary skill.
- **Q:** What’s my grade?  
**A:** The exact answer is unknowable until the end of the semester. For an approximate answer, compare your scores to the class distribution.
- **Q:** Can I have an extension on an assignment?  
**A:** In case of extenuating circumstances (medical issue, personal emergency, etc.), of course; please contact the instructor/TA. In exceptional cases, it may be appropriate to extend a deadline for the entire class, but such extensions may also be unfair to students who completed work by the original deadline. (see “Fairness”). If you think there is reason to extend a deadline for the class, please make the request well in advance of the deadline.
- **Q:** How can I improve my grade?  
**A:** Participate in lecture, come to office hours, study, ask questions, and start assignments **early**. Score changes after-the-fact are not possible, i.e., do not ask about grade improvements after the semester has ended but instead prepare throughout the semester so you can best achieve the course’s learning outcomes. See sections: “Fairness” and “Grading Corrections”.
- **Q:** Why are exam scores so “low”? (Half the class “failed!”)  
**A:** This course does not use an 90/80/70-percent scale (so half the class did not “fail”). Such a scale is (1) arbitrary and (2) poorly-aligned with open-ended and challenging nature of upper-level and graduate courses such as this one. In particular, rubrics for problem solving and proofs (both of which are a focus in this course) do not align well with a 90/80/70 scale, so lower scores do not necessarily indicate “failure.” Rather, this course is graded on a curve based on the statistical distribution of course scores and student effort.
- **Q:** Why does this course grade on curve?  
**A:** Curving supports *robust* determination of letter grades that are *fair* and *consistent*. Specifically:
  - The open-ended and challenging nature of assessments in an upper-level course results in a wider distribution of scores than low-level courses that evaluate more limited outcomes (i.e., the lower levels of Bloom’s taxonomy). Curving accommodates this wider distribution to produce grades that reflect learning outcomes.
  - Average scores change slightly over different terms, e.g., based on variations in difficulty of exam questions. Curved grading ensures that letter grades remain consistent.
- **Q:** Will you round scores?  
**A:** Generally, there is no rounding (modulo floating point error). I try to set the letter grade cutoffs where there are gaps in the scores so that students with nearly identical numerical scores do not end up with different letter grades.

- **Q:** Why does this course use. . .

- **Q:** . . . Git and Github?

- A:** In previous years, when students submitted tarballs on Canvas, they often struggled to share code with each other, and groups occasionally submitted incorrect versions of their project (resulting in much lower scores than the group expected!). Git (and Github) are critical tools to collaborate on code and to reduce the chance of submitting an unintended version. Moreover, Git is pervasive in professional software development.

- **Q:** . . . OCaml

- A:** OCaml is a functional programming language that is common in industry and offers a syntax similar to the textbook Lambda Calculus (the mathematical foundation of programming). Thus, OCaml aligns well with the *theory & practice* philosophy of Mines.

- **Q:** . . . Linux?

- A:** Primarily, the programming tools used in the course are best supported on Linux. Secondly, the instructor is unable to provide support for non-Linux systems (limited support for unix-like systems such as Mac OSX may be possible). Additionally, Linux proficiency—though not explicitly a learning outcome of this course—is vital for computing professionals, given the pervasive use of Linux in mobile devices, cloud computing, high performance computing, robotics, etc.

## A Proof Rubric

This course includes proofs. We will use the following rubric to grade proofs.

### Logic and Reasoning: 50%

- 0%: Totally wrong, all steps in the argument are illogical.
- 10%: At least one step is valid, but the argument is mostly wrong.
- 20%: Partially valid argument, but significant errors in the steps or conclusions.
- 30%: Overall argument is generally valid, but major errors present.
- 40%: Conclusion is valid, but minor errors present.
- 50%: All steps are correct and the argument is valid.

### Communication – Terminology and Notation: 20%

- 0%: Major errors in terminology indicating a lack of understanding.
- 10%: Minor errors in terminology or imprecise/informal language.
- 20%: Correct, formal, and precise use of terminology.

### Communication – Structure: 20%

- 0%: Difficult or impossible to follow.
- 10%: Follows basic structure for the type of proof.
- 20%: Flows well, suitable concise, and easy to read.

### Communication – Grammar: 10%

- 0%: Grammar problems impede understanding.
- 10%: Sufficiently correct grammar. No issues that affect understanding.