COMP 410: Logic Programming
Fall 2025

Instructor: Kyle Dewey (kyle.dewey@csun.edu)
Course Web Page: https://kyledewey.github.io/comp410-fall25
Office: JD 4419

Course Description (from the catalog)
Programming techniques in the logic programming language Prolog. Prenex conjunctive normal form and grammatical algorithms. Tableaux, sequenzen, resolution and other semi-decision procedures. Closures of relations, fixed point theory, control mechanisms, relationship to functional programming.

Learning Objectives
Successful students will be able to:
- Recognize problems which are well-suited to the logic programming paradigm
- Write Prolog programs which manipulate lists, solve NP-complete problem instances, and automatically generate software tests
- Understand how to work with logic programming features without logic programming languages, and effectively write logic programs in Python

Course Background, Emphasis, and Design
Logic programming, while often understated, is a major programming paradigm. In my opinion, logic programming is well-suited to problems with one or more of the following properties:
- There are many distinct answers of interest
- Solving the problem requires trying different approaches and seeing which works
- Answers can be described easily, but it's difficult to formulate how to arrive at an answer

This course emphases the use of logic programming to solve problems, along with programming techniques which are unique to the logic programming paradigm. To this end, there will be a multitude of programming assignments, most of which require you to write code. This code will be written in either Python or Prolog; Python serves as a sort of "typical" programming languages for our purposes, whereas Prolog is a classic logic programming language.

Will you ever use Prolog ever again? Honestly, probably not. However, my goal isn't to teach Prolog, but rather the ideas behind Prolog. Logic programming techniques can be implemented in more mainstream, non-logical languages; it's merely inconvenient opposed to impossible (similarly, object-oriented programming can be done in C, even though C predates object-oriented programming).

Textbook and Other Required Class Materials

No textbook is required. However, there are several sources which may be helpful:

- Learn Prolog Now! (http://www.learnprolognow.org/lpnpage.php?pageid=online); free online textbook that goes over the basics of Prolog
- The Art of Prolog (https://mitpress.mit.edu/books/art-prolog); very complete book on Prolog and logic programming which covers both basic and advanced topics; I have a copy if you'd like to peruse it
- The Craft of Prolog (https://mitpress.mit.edu/books/craft-prolog); discusses advanced topics in Prolog, including logic programming design patterns; I have a copy if you'd like to peruse it
- The Practice of Prolog (https://mitpress.mit.edu/books/practice-prolog); discusses real-world Prolog applications; I have a copy if you'd like to peruse it
- Logic, Programming, and Prolog (https://www.ida.liu.se/~ulfni53/lpp/bok/bok.pdf); free online textbook that focuses on the theoretical underpinnings of Prolog

Additionally, a computer, be it a laptop or otherwise, is required.

Grading

Your grade is based on the following components:

Assignments	15%
Midterm Exam 1	20%
Midterm Exam 2	30%
Final Exam	35%

Not all of these will be necessarily weighted evenly, nor will you necessarily be given the same amount of time for assignments. Exactly which assignments are assigned depends on how the class progresses. In general, assignments will be submitted through Canvas (https://canvas.csun.edu/). In the event that there is a problem with Canvas, you may email your assignment to me (kyle.dewey@csun.edu) as a last resort.

Plus/minus grading is used, according to the scale below:

If your score is >=...	**...you will receive...**
92.5	A
89.5	A-
86.5	B+
82.5	B
79.5	B-
76.5	C+

If your score is >=...	...you will receive...
72.5	C
69.5	C-
66.5	D+
62.5	D
59.5	D-
0	F

If you are not present for the final exam and you have not previously made alternative arrangements with me for the final exam, a grade of WU (unauthorized withdrawal) will be assigned.

Collaboration for Assignments
All students are required to submit their own individual work. For assignments (and **only** assignments), students may discuss among each other, as long as they don't digitally share code. That is, you **cannot** simply email your code to someone else. However, you **may** discuss your actual code with someone else, including merely viewing code. The only stipulation is that **if you do discuss with someone else, say so in your submission.** This is not for punitive reasons; this is only so I get a sense of who is working with who. My intention with this policy is to enable collaborative learning, as opposed to simply sharing a solution.

Plagiarism and Academic Honesty
While collaboration is allowed on assignments, you are responsible for all of your own work. You may **not** take code from online sources and submit it as your own. If you must take code from online, cite where you took the code from. Worst-case scenario, you'll receive a 0 for whatever you took, but no further action will be taken. In general, code taken online which solves more general things (e.g., "how do I iterate through an array in Java") is more acceptable than code which solves more specific things (e.g., "how do I implement a recursive find function over immutable linked lists in Swift"). General bits of code only give you pieces of a solution, whereas specific bits of code often will give you a complete copy/pastable solution. If it's not 100% clear if something is permitted to be used or not, you can always ask me beforehand.

Chegg is specifically disallowed as an online resource, as it's almost always used as a repository of complete questions with answers. That is, the questions/answers are practically always of the specific kind mentioned above, with zero learning whatsoever.

For assignments, LLM-based tools like ChatGPT are discouraged, though not outright disallowed. It can be easy to overly on such tools without realizing it, to the point where the tool does all the work, with zero understanding of what it produces. If you do use

such tools, you should be aware of the following trap I've seen a lot of students fall into over the past year:

- The student gets a solution from an LLM
- The student tests the solution, and observes that all tests pass
- The student looks at the code a bit, and decides that it "feels" right. This feeling is usually based on the relative length, complexity, formatting, and variable naming of the code, and not on its actual behavior.
- The student bombs a high-scoring, closely-related exam question, which asked them to write similar code for a similar problem.
- The student realizes they did not actually understand what the LLM produced, and then needs additional help to hopefully catch up.

In general, for this course, my opinion is that if an LLM is to be used, it's best to use prompts that are general (e.g., how does recursion in Prolog work?) as opposed to specific (e.g., write a Prolog program using recursion that solves the specific problem I've been asked to solve). A course-specific pitfall is that there is far less Prolog in LLM training sets than for more popular languages, and so LLMs have a tendency to produce incorrect code, especially for atypical problems.

No discussion whatsoever is allowed during exams, except with the instructor. Any violations can result in a failing grade for the assignment/exam, or potentially failing the course for egregious cases. A report will also be made to the Dean of Academic Affairs. Students who repeatedly violate this policy across multiple courses may be suspended or even expelled.

Communication
In general, any questions should be made through Canvas. You can also email me, though I'm usually much faster to respond to Canvas than my general email.

Late Policy / Exam Scheduling
Late assignments will be accepted without penalty if prior arrangements have been made or there is some sort of legitimate emergency (at my discretion). If you must be absent from an exam, contact me ASAP to see if alternative accommodations can be made.

If an assignment is otherwise submitted late, it will be penalized according to the following scale:

If your assignment is late by <= this many days...	...it will be deducted by...
1	10%
2	30%
3	60%

If your assignment is late by <= this many days...	...it will be deducted by...
4+	100%

To be clear, assignments which are submitted four or more days beyond the deadline will not receive credit. The reason for such a harsh late policy is that we will generally discuss solutions in class shortly after the deadline, and this late policy discourages people from simply pulling a solution from an in-class discussion.

Class Feedback

I am open to any questions / comments / concerns / complaints you have about the class. If there is something relevant you want covered, I can push to make this happen. I operate off of your feedback, and no feedback tells me “everything is ok”.

Class Schedule and List of Topics (Subject to Change)

Week	Monday	Wednesday
1	8/25: Introduction, motivation	8/27: SAT, semantic tableau
2	~~9/1: Labor day (no class)~~	9/3: SAT, semantic tableau
3	9/8: SAT, semantic tableau, abstract syntax trees	9/10: Abstract syntax trees, introduction to Prolog, nondeterminism, unification
4	9/15: nondeterminism, unification	9/17: Unification with structures, recursion
5	9/22: Unification with structures, recursion	9/24: Unification with lists
6	9/29: Unification with lists, recursion with lists	10/1: Unification with lists, recursion with lists
7	10/6: Midterm exam 1 review	10/8: **Midterm Exam 1**
8	10/13: Midterm exam 1 retrospective, generating tests	10/15: Context-free grammars, generating tests
9	10/20: Generating tests	10/22: Generating tests
10	10/27: Generating tests, optimizing Prolog code	10/29: Generating tests, optimizing Prolog code, Prolog control structures

Week	Monday	Wednesday
11	11/3: Prolog control structures, nondeterminism via iterators	11/5: Nondeterminism via iterators, nondeterminism in Python
12	11/10: Nondeterminism in Python	11/12: Nondeterminism in Python
13	11/17: Nondeterminism in Python, midterm exam 2 review	11/19: **Midterm Exam 2**
14	11/24: Midterm Exam 2 Retrospective, more nondeterminism in Python	11/26: More nondeterminism in Python, implementing unification in non-logical languages
15	12/1: Implementing unification in non-logical languages	12/3: Combined unification and nondeterminism in non-logical languages
16	12/8: Combined unification and nondeterminism in non-logical languages	12/10: Final Exam Review

Final exam: Monday, December 15 from 3 - 5 PM in JD 3510