COMP 410: Logic Programming
Fall 2021

Instructor: Kyle Dewey (kyle.dewey@csun.edu)
Course Web Page: https://kyledewey.github.io/comp410-fall21
Office: JD 4419 (I will not physically be there); Zoom link for office hours posted on Canvas and available via email.

Special COVID-19 Message
The course is being run as a synchronous online course. We will never meet in person, though we will meet virtually via Zoom regularly each week at the assigned time (see SOLAR for meeting time details). Assuming you're enrolled in the course or on the waitlist, I have emailed the Zoom link to you. If somehow you do not have the Zoom link, email me at kyle.dewey@csun.edu. For exams, if you cannot be on Zoom during the scheduled exam time, let me know ahead of time for alternative arrangements.

The synchronous lectures will be recorded and made available through Canvas, in case students want to view them asynchronously. These lectures will only be accessible to CSUN students either enrolled or waitlisted for the course. However, **by enrolling in, or waitlisting this course, you consent to having any voice or webcam recorded.** That said, **I will never require you to use your webcam or speak**; only what you voluntarily send will be recorded. Questions can either be asked verbally or textually through the Zoom chat. I will verbally repeat any questions in the chat before answering them, but I will not identify who said the question. Even so, it's possible that your name will end up in the meeting recording, identifying you as a participant (Zoom sometimes unavoidably shows participant names in the recording).

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 and Mercury programs which manipulate lists, solve NP-complete problem instances, automatically generate software tests, and interpret other Prolog programs
- Understand the connection between formal logic and logic programming, and the theoretical underpinnings of logic programming

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 Prolog or Mercury, both of which are reputable logic programming languages.

Will you ever use Prolog or Mercury ever again? Honestly, probably not. However, my goal isn't to teach these languages, but rather the ideas behind these languages. 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	30%
Midterm Exam 1	15%
Midterm Exam 2	25%
Final Exam	30%

Not all assignments will be weighted evenly, nor will you always 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), though this should be considered 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+
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.

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.

Attendance
In the first week of class, I will take attendance. If you miss both sessions in the first week and have not made alternative arrangements with me, you must drop the class, as per University policy (http://catalog.csun.edu/policies/attendance-class-attendance/). After the first week I will not take attendance, though you are strongly encouraged to attend.

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%

If your assignment is late by <= this many days...	...it will be deducted by...
2	30%
3	60%
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	Tuesday	Thursday
1	8/31: Introduction, motivation, functional programming refresher	9/2: functional programming refresher, BNF review, abstract syntax trees, SAT, semantic tableau
2	9/7: SAT, semantic tableau	9/9: SAT, semantic tableau, Introduction to Prolog
3	9/14: Nondeterminism, backtracking, recursion	9/16: Recursion, structures
4	9/21: Recursion, structures, unification	9/23: Structures, unification, lists
5	9/28: Lists, recursion with lists	9/30: Recursion with lists, review
6	10/5: **Midterm Exam 1**	10/7: Midterm exam retrospective, generating tests
7	10/12: Generating tests	10/14: Generating tests
8	10/19: Generating tests, optimizing Prolog code	10/21: Prolog-based interpreters

Week	Tuesday	Thursday
9	10/26: Prolog-based interpreters, metainterpreters	10/28: Metainterpreters
10	11/2: Metainterpreters, Prolog control structures	11/4: Metainterpreters, Prolog control structures
11	11/9: Peano arithmetic, constraint logic programming	~~11/11~~: Veteran's day (no class)
12	11/16: constraint logic programming, modes, review	11/18: **Midterm Exam 2**
13	11/23: Midterm exam retrospective, introduction to Mercury	~~11/25~~: Thanksgiving (no class)
14	11/30: Types and data structures in Mercury	12/2: Types and data structures in Mercury, higher-order procedures
15	12/7: Higher-order procedures	12/9: Final exam review

Final exam: Tuesday, December 14 from 3 - 5 PM