CSc 372 — Comparative Programming Languages
Spring 2026 (McCann)

https://cs.arizona.edu/classes/cs372/spring26/

Program #3: Prolog

Due Date: April 27th, 2026, at the beginning of class

Overview: Easy: Time to add "doing with Prolog" to "talking about Prolog."

General Requirements: The following requirements apply to all parts of this assignment:

1. You are to use SWI Prolog (`swipl` on lectura, or you can install it on your own computer). Be sure that you test your predicates on lectura, as that's where the TAs will be grading them.
2. Provide good documentation for all of your predicates. Remember, *you* may be the future person who needs to understand how they work!
3. For all tasks:
   (a) You may write helper predicates if you'd like to do so. Sometimes, you won't have a choice.
   (b) Tasks may have restrictions on what you may and/or may not use from Prolog's provided features. If there are no restrictions, you may use any features you wish.
   (c) If your predicate doesn't stop after finding the correct answer, that's OK. Just press ENTER (instead of the semicolon key) to stop it.
   (d) Because of Prolog's nature, predicates are not expected to 'sanity–check' the provided arguments. If your predicate is invoked with garbage, the invoker deserves what they get.
4. **AI Tool Usage:** You may use AI tools (LLMs, Generative AIs, IDE AI plug–ins, etc.) to help you write code on the programming assignments in this class. However, we *strongly* recommend that you use it to help you only when you get stuck, because you will be asked to write code on the exams. For that reason, write the code yourself, as much as possible, to help you learn the languages covered in this class.

Assignment: For each numbered task below, write a complete, well–documented Prolog predicate (named as stated) that accomplishes it.

1. **Surface Area and Volume of Spheres**.
   - File name: `sphere.pl`
   - Predicates: `spherearea(radius,area)`, `spherevolume(radius,volume)`.
   - Sample Queries:

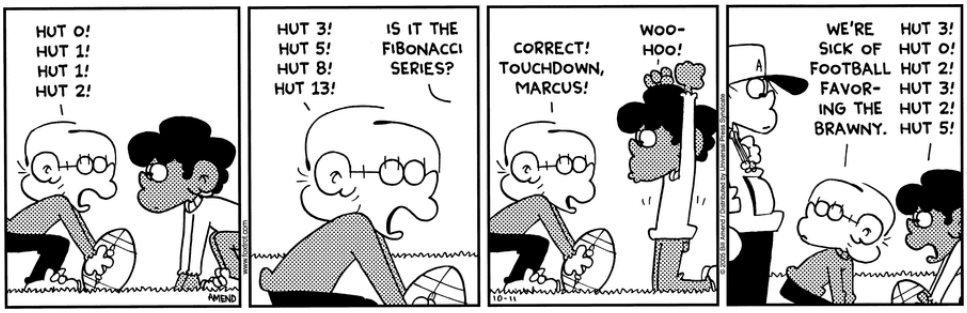
```
?- spherearea(5,X).
X = 314.1592653589793.
?- spherevolume(5,X).
X = 523.5987755982989.
```

   The formulae for the (surface) area and volume of a sphere are $4\pi r^2$ and $\frac{4}{3}\pi r^3$, respectively. Notice that the latter is a small extension of the former. Take advantage of this: Write the `spherevolume` predicate to invoke the `spherearea` predicate.

   Use Prolog's built–in `pi/0` function for π's value.

(Continued...)

2. **Perrin Sequence**.



- File name: `perrin.pl`
- Predicate structure: `perrin(seqnum,seqvalue)`
- Sample Queries:

```
?- perrin(0,X).
X = 3.
?- perrin(10,X).
X = 17.
```

In class we examined two ways to compute members of the Fibonacci sequence. A similarly–defined sequence is the Perrin Sequence, the first six members of which are listed in the October 11, 2015, Foxtrot strip shown above.[1]

The recurrence relation that defines the Perrin sequence is: $p(n) = p(n-2) + p(n-3)$, where $p(0) = 3, p(1) = 0$, and $p(2) = 2$. Using this recurrence, write the `perrin` rule.

3. **Remainder of Integer Division via Subtraction**.

- File name: `leftover.pl`
- Predicate structure: `leftover(dividend,divisor,remainder)`
- Sample Queries:

```
?- leftover(15,4,X).
X = 3 .
?- leftover(6,7,X).
X = 6 .
?- leftover(12,0,X).
ERROR: leftover/3: Arithmetic: evaluation error: ‘zero_divisor’
true .
```

You know how integer division works. Write `leftover` so that the third argument returns the remainder of the integer division of the dividend by the divisor. If the divisor is zero, use the `write/1` predicate to display the same error message (with an updated predicate name) that SWI Prolog produces when 1/0 is attempted.

Restriction: As the title says, we expect you to write `leftover` to use repeated subtractions to discover the remainder. Thus, built–in predicates such as `//`, `div/2`, and `rem/2` cannot be used in your rule(s).

(Continued...)

[1] http://www.gocomics.com/foxtrot/2005/10/11

4. **Print Second, Fourth, Sixth, etc., Elements of a List**.

   - File name: `print246etc.pl`
   - Predicate structure: `print246etc(list)`
   - Sample Queries:

```
?- print246etc([]).
true.
?- print246etc([j]).
true.
?- print246etc([k,l,m,n,o,p,q]).
l
n
p
true.
?- print246etc([r,s,t,u]).
s
u
true.
```

   Our first list predicate. The concept is straight–forward. You'll want to use `write/1` to display the list elements. Not much else to say!

5. **Delete from List (Repeatedly)**.

   - File name: `delfromlist.pl`
   - Predicate structure: `delfromlist(element,list,result)`
   - Sample Queries:

```
?- delfromlist(1,[1],X).
X = [].
?- delfromlist(a,[a,b,r,a,c,a,d,a,b,r,a],X).
X = [b, r, c, d, b, r].
```

   `delfromlist` deletes all occurrences of a constant (the first argument) from a list (the second argument), returning the (potentially) shortened list (the third argument). Note that the deletions only need be performed at the top–level of the list; that is, if the list contains lists, `delfromlist` is not expected to examine the content of those contained lists.

   Restriction: Predicates such as `delete/3` are off–limits.

(Continued...)

6. **A Subset of All Sets?**

   - File name: `subsetofall.pl`
   - Predicate structure: `subsetofall(subset,listofsets)`
   - Sample Queries:

```
?- subsetofall([],[]).
false.
?- subsetofall([],[[]]).
true.
?- subsetofall([d],[[c,d],[d,e],[e,f]]).
false.
?- subsetofall([m,n],[[n,m],[l,m,n],[m,n,o]]).
true.
```

   The idea: Given a set, and a set of sets, is the first set a subset of all of the sets in the set of sets? Write `subsetofall` to accomplish this task.

   Restriction: SWI Prolog has `subset/2` and `ord_subset/2`. You may not use either, or any other built-in subset predicate.

   Hints: (1) Because you'll need to test the given set against each of the sets in the list of sets, it will be useful to have a helper predicate that checks to see if one set is the subset of another set. (2) You'll find `member/2` to be useful in writing that helper. We'll be spending a decent amount of time in class understanding how `member/2` works.

Data: The input expectations for the predicates are given with the predicate descriptions, above.

Output: Output details, when a task has any, are stated in the Assignment section, above. Otherwise, the tasks expect a variable to be supplied that will represent the product of the predicates.

Hand In: You are required to submit your completed program files (your `.pl` files) using the `turnin` facility on lectura. The submission folder is `cs372p3`. Instructions are available from the document of submission instructions linked to the class web page. In particular, because we will be grading your program on lectura, it needs to run on lectura, so be sure to test it on lectura. Submit all files as-is, *without* packaging them into `.zip`, `.jar`, `.tar`, etc., files.

Additional Hints, Reminders, and Requirements:

- There is a list of Prolog resources on the class web page. As with Ruby and Haskell, working through statement syntax, execution errors, etc., are chores that we expect you to try hard to do on your own, as you would have to if you were learning a language on your own. But, if you get really stuck, the TAs and I will try to help.
- We realize that we haven't yet demonstrated all of the features of Prolog that you will need to complete all of these predicates. You can either wait for those features to be covered, or you can explore them on your own. We strongly encourage the latter!
- **Start Early!** This assignment overlaps with the final project (by design). The earlier you start, the sooner you'll finish, and the more time you'll have to work on the project. We don't expect that any of these predicates will be *conceptually* difficult for you, but they will probably take some time to complete, due to your inexperience with the language.