Recap
What are the four types of terms in Prolog?

Atoms, variables, numeric literals, and structures.
How do you declare a variable in Prolog?

You don’t, you just refer to it.
How can you tell atoms and variables apart?

Variable names start with a capital letter, atoms either start with a lower-case letter or are quoted.
Write a Prolog clause `xor_equal/4` that has the following behavior:

\[
\begin{align*}
xor\_equal(a, a, b, c). & \rightarrow \text{true.} \\
xor\_equal(a, b, b, b). & \rightarrow \text{true.} \\
xor\_equal(a, a, b, b). & \rightarrow \text{false.}
\end{align*}
\]

\[
\begin{align*}
xor\_equal(A, B, C, D) & :\!- A = B, C = D, !, \text{fail.} \\
xor\_equal(A, B, \_, \_) & :\!- A = B, !. \\
xor\_equal(\_, \_, C, D) & :\!- C = D.
\end{align*}
\]
How can you do iterative computations in Prolog?

Only with recursion, since there are no loops.
Is this a $LL(1)$ grammar?

\[
\text{foo} \rightarrow x \mid y \mid z \text{ foo} \mid \text{ bar foo baz} \mid \text{ foo foobar bar}
\]

\[
\text{foobar} \rightarrow 1 \mid 2 \mid 3
\]

No, \textit{foo} is left-recursive.
Write a Prolog clause `count/0` that outputs the numbers 1 through 100.

Hint:
output a term with `write/1`, cause a line break with `nl/0`.

count :- count(1).
count(101).
count(X) :-
    write(X), nl,
    Y is X + 1,
    count(Y).
How can we implement a linked list in Prolog?

See sample code…