The natural numbers are built from two constructs, the constant symbol 0 and the successor function s. All the natural numbers are then recursively given as 0, s(0), s(s(0)), ..., such that:

number(0).
number(s(X)) :- number(X).

Having defined numbers, we can proceed to define addition. Since ∀x, 0 + x = x, we have

plus(0, X, X) :- number(X).

1. Write 1 + 2 = 3 as a Prolog ground fact.
   Solution:
   plus(S(0), S(S(0)), S(S(S(0)))).

2. Create a new rule of inference for a general plus operation. (Hint: Think recursion)
   Solution:
   plus(S(X), Y, S(Z)) :- plus(X, Y, Z).

3. Use the above addition rule to write leq(X, Y) and lt(A, B) such that X ≤ Y, A < B
   Solution:
   leq(X, Y) :- plus(X, T, Y).
   lt(X, Y) :- plus(X, S(T), Y).

4. Building on top of addition, let’s write a set of rules that describe multiplication:
   times(X, Y, Z) means XY = Z
   Solution:
   times(X, 0, 0) :- number(X).
   times(X, S(Y), Z) :- plus(R, X, Z), times(X, Y, R).
5. Write \( \text{mod}(X, Y, Z) \) such that \( X \mod Y = Z \)

Solution:

\[
\begin{align*}
\text{mod}(X,Y,X) & :\ :- \lt(X,Y). \\
\text{mod}(X,Y,Z) & :\ :- \text{plus}(X_1,Y,X), \text{mod}(X_1,Y,Z).
\end{align*}
\]