A pure logic programming language does not incorporate any form of arithmetic - we have to build our own arithmetic directly from logical concepts. Firstly we must build a representation of the concept of number and then tools for manipulating numbers.

Numbers will be represented by terms involving the constant zero and a function $s$ which will represent the successor function (i.e. the function $s(x) = x+1$). Therefore the terms zero, $s($zero$)$, $s(s($zero$))$, $s(s(s($zero$)))$ etc. will represent the positive integers.

We define a Prolog predicate that can be used to recognize or generate terms that are numbers.

\[
\begin{align*}
isnumber(zero). \\
isnumber(s(X)) & :\text{isnumber}(X). 
\end{align*}
\]

Having this definition we can now define some standard predicates on our set of numbers.

\[
\begin{align*}
isequal(X,X) & :\text{isnumber}(X). \\
isequal(s(X),s(Y)) & :\text{isequal}(X,Y). \\
lessthanequal(zero,X) & :\text{isnumber}(X). \\
lessthanequal(s(X),s(Y)) & :\text{lessthanequal}(X,Y). 
\end{align*}
\]

Experiment with these predicates and see how they work. Can they be used in more than one fashion (try querying the database with the arguments being variables for example).

We can also define the arithmetic operations by thinking about them in a logical fashion. For example addition can be defined logically in the following fashion.

zero added to anything yields the same thing
if $x + y = z$ then $(x+1) + y$ must equal $z + 1$

which translates into Prolog as

\[
\begin{align*}
add(zero,X,X) & :\text{isnumber}(X). \\
add(s(X),Y,s(Z)) & :\text{add}(X,Y,Z). 
\end{align*}
\]

Experiment with this definition. For the rest of your lab time try writing the following predicates

$odd(X)$ to be true if $X$ is odd
$even(X)$ to be true of $X$ is even
times(X,Y,Z) to be true if \( X \times Y = Z \)

\( \text{quotient}(X,Y,Q) \) to be true if \( X/Y = Q \) (in integer arithmetic)

\( \text{remainder}(X,Y,R) \) to be true if \( X \) divided by \( Y \) leaves a remainder of \( R \)

\( \text{fact}(N,X) \) to be true if \( X = N! \)

\( \text{fibonacci}(N,X) \) to be true if \( X \) is the \( N \)'th Fibonacci number

Define now a predicate \( \text{shownum}(X,N) \) which is true when symbolic \( X \) corresponds to the natural number \( N \) (e.g. \( \text{shownum}(s(zero), 1) \) is true). Check the following queries:

\[
\text{?-shownum}(s(s(zero)),X).
\]
and

\[
\text{?-shownum}(Y,5).
\]

Hint: use SWI-Prolog in-built arithmetic.

Finally verify the previously defined predicates (e.g. \( \text{times}(X,Y,Z) \)) by using predicate \( \text{shownum}(X,N) \).

Think how to define predicates \( \text{quotient}(X,Y,Q) \) and \( \text{remainder}(X,Y,R) \) for \( Y = \text{zero}! \) (should you have not done it already).

Assoc. Prof. R. Kozera