1. Write a predicate that has as input an 
s-number \( I \) - like \((s\,(s\,(s\,(0)))\)) 
a) is true if \( I \) is even 
b) is true if \( I \) is odd

2. Write a predicate \texttt{less(+X, +Y)} that 
- assumes that \( X \) and \( Y \) are s-numbers, and 
- is true if \( X < Y \).

3. Write a predicate \texttt{to_zero(+I, ?L)} 
- that assumes that \( I \) is an s-number, and 
- is true if \( L \) is the list \([I, \ldots, 0]\).

4. Write a predicate \texttt{scale(+Xs,+M,-Ys)} 
- that assumes that \( Xs \) is a list of s-numbers 
  and \( M \) is an s-number; 
- is true if \( Ys \) is the list obtained by 
  multiplying each element of \( Xs \) by \( M \). 
Refer to \texttt{times/3} for the multiplication.

Compare to functional programming (SML): 
\[
\begin{align*}
\text{fun scale(nil,\_)} &= \text{nil} \\
\text{scale(x::xs,m)} &= x\times m :: \\
& \quad \text{scale(xs,m)};
\end{align*}
\]
Now we start to use Prolog’s built-in numbers.

5. Translate the following program to Prolog:
   
   ```prolog
   while i >= 1 do
       writeln(i);
       i := i-1
   od
   ```

   to a program that
   - has as input a number \( I \)- like 3
   - prints the numbers \( I, \ldots, 1 \)

   Translate `writeln(i)` to `write(I),nl`

   To compute \( I-1 \), write `J is I-1`.

   Comment: \( I \) is \( I-1 \) is just not true!

6. Write a program that returns the maximum of a nonempty list of numbers.
   * Use a help-predicate that has the maximum so far as input:
   `max(+RestOfList,+MaxSoFar,-Max)`

7. If you are bored: read § 3.5 of the compendium and try the exercise.