Hamming's exercise in SASL.

In a slightly simplified version, the problem is to generate in increasing order "all" positive integers that have no other prime factors than 2 and 3. The solution a la SASL that was designed by J. L. A. van de Snepscheut was essentially the following:

\[
\begin{align*}
\text{def } & \quad \text{times } n \ (a; b) = n \times a \ : \ \text{times } n \ b \ ; \\
\text{def } & \quad \text{merge } (a_0; b_0) \ (a_1; b_1) = \\
& \quad \text{if } a_0 < a_1 \rightarrow a_0 : \ \text{merge } b_0 \ (a_1; b_1) \\
& \quad \text{if } a_0 = a_1 \rightarrow a_1 : \ \text{merge } b_0 \ b_1 \\
& \quad \text{if } a_0 > a_1 \rightarrow a_1 : \ \text{merge } (a_0; b_0) \ b_1 \\
\text{def } & \quad x = 1 : \ \text{merge } (\text{times } 2 \ x) \ (\text{times } 3 \ x); \\
\text{X } & \quad \text{(0)}
\end{align*}
\]

This note records how — at the present state of my art — I prove the correctness of this program.

* * *

I need a notation for the n-th element of a continued concatenation:

\[
\text{sub } n \ (a; b) = \begin{cases} 
n = 0 & \rightarrow a \\
 n > 0 & \rightarrow \text{sub } (n-1) \ b \\
\end{cases}
\]

I need it in order to be able to express a simple
Theorem about the function dist which—remember the low binding power of the concatenation as indicated by the colon—is given by

\[ \text{dist} \ f \ (a; b) = f \ a : \text{dist} \ f \ b. \]

**Theorem 0:**

\[ (\forall n : n > 0 : \text{sub} \ n \ (\text{dist} \ f \ x) = f \ (\text{sub} \ n \ x)) \]

**Proof of Theorem 0.** By mathematical induction (by what else?)

**Case** \( n = 0 \):

\[ \text{sub} \ 0 \ (\text{dist} \ f \ (a; b)) = \]

\[ \text{sub} \ 0 \ (f \ a : \text{dist} \ f \ b) = \]

\[ f \ a = \]

\[ f \ (\text{sub} \ 0 \ (a; b)) \]

**Case** \( n > 0 \):

\[ \text{sub} \ n \ (\text{dist} \ f \ (a; b)) = \]

\[ \text{sub} \ n \ (f \ a : \text{dist} \ f \ b) = \]

\[ \text{sub} \ (n-1) \ (\text{dist} \ f \ b) = \]

\[ f (\text{sub} \ (n-1) \ b) = \]

\[ f (\text{sub} \ n \ (a; b)) \]

(End of Proof of Theorem 0).

I think Theorem 0 so trivial that most people are willing to apply it without bothering to think about its proof. It should be formulated and proved once.
Let \( \min s \) be the smallest element from a set \( s \) of natural numbers. For an infinite set \( s \) we define the continued concatenation of \( s \) by

\[
sort s = \min s : \text{sort} \left( s - \{ \min s \} \right).
\]

In terms of the set \( \text{formator} \cup \) we have to prove that in (0)

\[
x = \text{sort} \left( \cup n_2, n_3 : n_2 \geq 0, n_3 \geq 0 : 2^{n_2} \cdot 3^{n_3} \right) \tag{1}
\]

To do so, we first prove about merge

**Theorem 1.** For two infinite sets \( S_0 \) and \( S_1 \) of natural numbers — the union of which we denote by \( S_0 + S_1 \) — we have

\[
\text{merge} \left( \text{sort } S_0 \right) \left( \text{sort } S_1 \right) = \text{sort} \left( S_0 + S_1 \right)
\]

**Proof of Theorem 1** The theorem to be proved is of the form

\[
(\forall S_0, S_1 : P(S_0, S_1) = Q(S_0, S_1))
\]

where \( P \) and \( Q \) are functions of two infinite sets of natural numbers; their values being continued concatenations we prove

\[
(\forall n : n \geq 0 : (\forall S_0, S_1 : \text{sub } n \left( P(S_0, S_1) \right) = \text{sub } n \left( Q(S_0, S_1) \right)))
\]

and this is proved using mathematical induction over \( n \). The proof is straightforward and left to the reader.

(End of Proof of Theorem 1.)
It is now not difficult to show that \( x \), as given by (1) satisfies (0). Thanks to Theorem 0 and \( p < q \Rightarrow 2p < 2q \), we find

\[
\text{times } 2\ x = \text{sort} \left( \bigcup n_2, n_3 : n_2 \geq 1, n_3 \geq 0 : 2^{n_2} \cdot 3^{n_3} \right)
\]

and, similarly,

\[
\text{times } 3\ x = \text{sort} \left( \bigcup n_2, n_3 : n_2 \geq 0, n_3 \geq 1 : 2^{n_2} \cdot 3^{n_3} \right)
\]

and then, thanks to Theorem 1

\[
\text{merge} \left( \text{times } 2\ x \right) \left( \text{times } 3\ x \right) = \text{sort} \left( \bigcup n_2, n_3 : n_2 \geq 0, n_3 ; n_2 + n_3 \geq 1 : 2^{n_2} \cdot 3^{n_3} \right)
\]

from which immediately follows

\[
1 : \text{merge} \left( \text{times } 2\ x \right) \left( \text{times } 3\ x \right) = x
\]

* * *

In order that program (0) is an effective program I think that we have to prove more. It seems that we have to prove the uniqueness of solution (1).

With uniq defined by

\[
\text{uniq} \ n \ x = (\forall i : 0 \leq i < n : \text{sub} \ i \ x \neq \omega)
\]

I think we should prove in succession

\[
\text{uniq} \ n \ x \Rightarrow \text{uniq} \ n \ \left( \text{times} \ m \ x \right)
\]

\[
\left( \text{uniq} \ n \ y \right) \land \left( \text{uniq} \ n \ z \right) \Rightarrow \text{uniq} \ n \ \left( \text{merge} \ y \ z \right)
\]

and, finally, from the last definition of (0)

\[
\text{uniq} \ n \ x \Rightarrow \text{uniq} \ (n+1) \ x
\]

These proofs are very easy, but my current feeling is that we are not allowed to omit them.
(Continued after walking with the dogs.) My choice of identifier "uniq" betrays my misunderstanding. Suppose the last definition had been
\[ x = \text{times} \ 2 \ x \] (2)
This – as is easily verified – has one solution, viz.
\[ x = \text{zeroes} \]
(with def zeroes = 0 : zeroes). The solution is unique, but I expect no implementation to find it. We fool ourselves when we regard a SASL program just as a set of recursive equations. They are recursive definitions, quite definitely with a direction! Perhaps it would have been more honest to write the definition
\[ \text{zeroes} := 0 : \text{zeroes} \]

Plataanstraat 5 15 June 1981
5671 AL NUENEN prof.dr. Edsger W. Dijkstra
The Netherlands Burroughs Research Fellow