Halt!
by L.K.

Theorem. In a given programming language L, there is no (algorithmic) way to list all the algorithms in L that halt.

Proof. Suppose that S={A1,A2,A3,...} is any (algorithmically produced) list of halting
algorithms written in the language L. We shall assume that each Ai accepts integers n as inputs and for each n, Ai(n) halts. Now define a new algorithm A as follows:

To run A(i):
1. Get a simulation of Ai.
2. Run Ai(i).

Now suppose that A = Aj for some j.
Then to run A(j), we must
1. Get a simulation of Aj.
2. Run Aj(j).

But Aj = A. So when we run Aj(j), we run A(j) and so this takes us back to step 1. In other words, the algorithm A will have an infinite loop if A is of the form Aj for any j. Therefore A is not on the list.
The list of halting algorithms is incomplete.
Q.E.D.

Note that this is exactly the same logic as:

A set X is well-founded (wf for short) if X has no infinite descending chains of membership.

Theorem. Any set of wf sets is incomplete. In particlular, if W is a set of wf sets, then W is itself well-founded and W is not a member of itself.

Proof. Any set W of wf sets is wf. For if we look for a descending chain of membership, we shall have to choose a member of W and look there. But that member is wf and so all chains of membership in that element terminate. W cannot be a member of itsself, for then W
would have an infinite descending chain of membership.
Q.E.D.