1. The Unsolvability of the Halting Problem (Turing, 1936)

We define two pieces of notation:

(1) '$P(A_1,\ldots,A_n)$' for 'procedure P applied to input $A_1,\ldots,A_n$'

(2) '#P' for 'description of (procedure) P'

Claim: There is no procedure P such that, for any procedure Q, and A any input to Q, P determines whether Q(A) halts or loops forever.

Proof: Suppose there were such a procedure. Call it 'HALT'. Let it have the following description (#HALT):

(3) HALT(#Q,A) = Output 'H' if Q(A) halts; then halt.
Output 'L' if Q(A) loops; then halt.

If HALT is a possible procedure, then so is HALT*:

(5) HALT*(#Q) = Output 'H' if HALT(#Q,#Q) outputs 'H'; then halt.
Output 'L' if HALT(#Q,#Q) outputs 'L'; then halt.

If HALT* is a possible procedure, then so is HALT**:

(6) HALT**(#Q) = Loop if HALT*(#Q) outputs 'H'.
Halt if HALT*(#Q) outputs 'L'.

But HALT** is not a possible procedure, for consider:

(7) HALT**(#HALT**) = Loop if HALT*(#HALT**) outputs 'H' and halts;
if HALT(#HALT**,#HALT**) outputs 'H' and halts;
if HALT**(#HALT**) halts.
X CONTRADICTION
Halt if HALT*(#HALT**) outputs 'L' and halts;
if HALT(#HALT**,#HALT**) outputs 'L' and halts;
if HALT**(#HALT**) loops.
X CONTRADICTION

More compactly:

(8) HALT**(#HALT**) loops if, and only if, HALT**(#HALT**) halts.

This is a contradiction. Hence, there is no procedure HALT. Q.E.D.

2. So What?

Turing's theorem is noteworthy for at least three reasons. (1) It is a <u>beautiful</u> argument. Although it establishes a <u>profound</u> result concerning <u>all</u> mechanical procedures, it relies on little more than the "ordinary" notion of a mechanical procedure. Naturally, a <u>rigorous</u> demonstration of the theorem, of the sort a mathematician demands, involves somewhat more technical machinery; but not <u>that</u> much more. (2) It is a <u>useful</u> result. It shows us that the class of formulable problems not solvable by purely mechanical means is not empty. It stimulates us to wonder what other problems, equally trivial in appearance, are unsolvable. (3) It is <u>philosophically significant</u>. In establishing a limit to mechanical procedures, it invites us to reconsider traditional philosophical puzzles about freedom and determinism, thought and mechanism.

3. The Halting Problem and Programming.

The proof that the Halting Problem is unsolvable can be expressed directly as a demonstration that a certain BASIC program cannot be devised.

We suppose that a program ("HALT") can be written in BASIC. HALT utilizes two input variables Q$ and A$, and produces a single output variable H$. HALT is defined:

(1) HALT(Q$,A$) produces H$, H$ = 'H' if the (BASIC) program whose text is in Q$ halts when applied to A$; else H$ = 'L'.

The actual program can be supposed to have the following structure:

```
100 GOSUB 400
110 IF H$ = 'H' THEN GOTO 110
120 END
400 LET A$ = Q$
410 GOSUB 500
420 RETURN
500 ...
    ...
  N RETURN
```

Lines 500-N contain the hypothetical HALT program. Lines 400-420 contain the analogue of HALT*. Lines 100-120 contain the analogue of HALT**.

The "paradoxical" nature of the program emerges when we consider its behavior under the following circumstances:

```
OK LET Q$ = '100 GOSUB 400;110 IF H$ = ''H'' THEN ...'
OK RUN
```

When started on Q$ = the text of itself, the program will halt only if it determines that it won't halt; it will loop only if it determines that it won't loop. Trouble. But not with lines 100-420. So, with lines 500-N, our hypothetical solution to the Halting Problem. But no more about that solution was said than that it <u>was</u> a solution. So no such solution can be given in BASIC.