If we now run *SELF* we observe the following behavior.

1. First A runs. It prints $\langle B \rangle$ on the tape.
2. B starts. It looks at the tape and finds its input, $\langle B \rangle$.
3. B calculates $q(\langle B \rangle) = \langle A \rangle$ and combines that with $\langle B \rangle$ into a TM description, $\langle SELF \rangle$.
4. B prints this description and halts.

We can easily implement this construction in any programming language to obtain a program that outputs a copy of itself. We can even do so in plain English. Suppose that we want to give an English sentence that commands the reader to print a copy of the same sentence. One way to do so is to say:

Print out this sentence.

This sentence has the desired meaning because it directs the reader to print a copy of the sentence itself. However, it doesn't have an obvious translation into a programming language because the self-referential word "this" in the sentence usually has no counterpart. But no self-reference is needed to make such a sentence. Consider the following alternative.

Print out two copies of the following, the second one in quotes:
"Print out two copies of the following, the second one in quotes:"

In this sentence, the self-reference is replaced with the same construction used to make the TM *SELF*. Part B of the construction is the clause:

Print out two copies of the following, the second one in quotes:

Part A is the same, with quotes around it. A provides a copy of B to B so B can process that copy as the TM does.

The recursion theorem provides the ability to implement the self-referential *this* into any programming language. With it, any program has the ability to refer to its own description, which has certain applications, as you will see. Before getting to that we state the recursion theorem itself. The recursion theorem extends the technique we used in constructing *SELF* so that a program can obtain its own description and then go on to compute with it, instead of merely printing it out.

THEOREM 6.3

Recursion theorem Let T be a Turing machine that computes a function $t\colon \Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$. There is a Turing machine R that computes a function $r\colon \Sigma^* \longrightarrow \Sigma^*$, where for every w,

$$r(w) = t(\langle R \rangle, w).$$

The statement of this theorem seems a bit technical, but it actually represents something quite simple. To make a Turing machine that can obtain its own description and then compute with it, we need only make a machine, called T