A. Prove that for any strings x and y, \( K(xy) \leq c + 2 \log_2 K(x) + K(x) + K(y) \). What exactly is c? (This problem is partially done in your book.)

B. We have proven (or will soon prove) in class that \( K(x) \) is not computable. So something must be wrong with the following argument. What’s wrong?

We will build a Turing machine \( N \) that computes \( K(x) \). Given an input x, \( N \) tests all strings y, in lexicographic order, to see whether they are descriptions of x. For each y, \( N \) first checks that y is of the form \( \langle M, w \rangle \). If it is, then \( N \) runs \( M \) on \( w \). \( N \) tests these strings y in parallel, in the usual way: one step on the first string, then two steps on the first string and one on the second, etc. As soon as \( N \) finds a string that describes \( x \), \( N \) halts with the length of that string on its tape.

This works because there is a bound on the length of string that \( N \) must try. Let \( M \) be the Turing machine that immediately halts, and let \( c = |\langle M, \rangle| \). Then, for any string \( x \), \( \langle M, x \rangle \) is a description of \( x \) of length \( c + |x| \), and so \( K(x) \leq c + |x| \). Thus \( N \) will find a description of \( x \) among the strings of length less than or equal to \( c + |x| \).

This last problem is harder. You probably won’t be able to do it until after we prove that \( K(x) \) is uncomputable. It’s very good practice for this theory, however.

C. Problem 6.23. (Hint: Mimic our proof that \( K(x) \) is not computable.)