Problem C. Edit Program

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Little Zhenya teaches her robot Murzik how to calculate the edit distance. As it is known, the edit distance, or Levenshtein distance, between strings s and t is the minimum possible number of actions required to transform s into t. Here, each of the actions must be either removing any single character from the string, inserting any single character into any position, or changing any single character into any other single character. For example, the edit distance between strings "`sport`" and "`point`" is three: we can remove letter "`s`", change "`r`" into "`i`", and insert "`n`" after it.

Murzik is still young, and only works well as long as the data is in binary: strings consisting of zeroes and ones. Still, a program for the robot can contain other characters too. Zhenya prepared a module for Murzik which will help him grasp the concept by working through examples. The module takes a string s of zeroes and ones as input data, and a string p as the program. The robot puts a pointer to the left edge of string s, before its first character, and runs the program step-by-step from left to right. The characters of the program correspond to steps as follows:

- "`.`": simply move the pointer one character to the right,
- "`x`": remove the character after the pointer,
- "`0`": insert character "`0`" at the pointer's position and put the pointer to the right of it,
- "`1`": insert character "`1`" at the pointer's position and put the pointer to the right of it,
- "`^`": change the character after the pointer into opposite one and put the pointer to the right of it.

When a character is removed, the remaining parts of the string move closer to each other so that there is no empty space left. When a character is added, the parts move away from each other to free a place for the new character. A program which acts with a character to the right of the pointer is invalid if there are no such characters.

Zhenya chose two binary strings s and t with the same length n, and now wants to construct a valid program such that, after running it on string s, Murzik will get string t, and the pointer will be located after the last character of the string.

All steps except simple pointer moves are actions from the definition of edit distance. A "`.`" step is not considered an action. For her example, Zhenya wants to construct any program which will have strictly less than n actions: minimizing the number of actions is not required. Surely, for short strings, finding such a program is easy as long as it exists, but Zhenya got thinking: how to construct such a program when the strings contain hundreds of thousands of characters?

Solve a generalized version of Zhenya's problem. Given binary strings s and t of the same length n, find any program which will allow Murzik transform s into t with strictly less than n actions, or determine that there is no such program. Remember that, after the program is run, the pointer must be located after the last character of the string.

Input

The first line of input contains a binary string s, and the second one contains a binary string t. It is guaranteed that s and t contain the same number of characters n ($1 \le n \le 10^6$).

Output

Print a single line. If there is no suitable program, the line must contain a single character "`-`" (ASCII code 45). Otherwise, print any suitable program consisting of characters "`.x01^`" (ASCII codes 46, 120,

48, 49, and 94). The number of actions must be strictly less than n, but is not required to be minimal possible. After the program is run, the pointer must be located after the last character of the string.

Example

standard input	standard output
0011010 1100100	xx...^10.
1 0	-