Too Many Hyphens

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Document markup languages such as TEX and LATEX, originally developed by Donald Knuth and Leslie Lamport, are often used in typesetting various articles, scientific texts, and even the statements for programming olympiads.

They provide special macros which allow one to use special characters that aren't present on the keyboard. For example, several hyphens in a row are converted into dashes of various lengths. Thus, when it's necessary to get a line of several hyphens in a document, you need to make some special actions.

In this case one can use braces to prevent a group of successive hyphens to be a macro. As in many programming languages (C++, for example) braces are used in TEX to group character blocks. Braces sequence should also form a correct parentheses sequence. That means that after removing all the characters from the line except braces, and then replacing the opening brace "`{`" with +1 and closing brace "`}`" with −1, the sum of all the elements of the resulting sequence should be equal to 0, and the sum of any prefix of the resulting sequence should be non-negative.

To prevent the sequence of hyphens from being replaced by a dash there should be at least one brace between any two consecutive hyphens. Note that the group of consecutive pluses, unlike hyphens, does not correspond to any macro and thus there are no extra conditions for two consecutive pluses.

Let's consider the string s consisting of pluses and hyphens. Let's add the minimum number of curly braces considering the rules described above, so that there are no two consecutive hyphens. Let's call the resulting string an *optimal escaped* string for s.

For example, for the string "`++--`" there are exactly five optimal escaped strings: "`++-{-}`", "`++-{}-`", "`++{-}-`", "`+{+-}-`", "`{++-}-`".

The lines above are listed in *lexicographical order*: they are ordered by the first unequal character (by the first, then by the second in case the first characters are equal, etc). Characters are ordered as "`+`" < "`-`" < "`{`" < "`}`".

Let's consider all the minimal length strings that can be obtained from s by adding bracers by the described rules, namely all optimal escaped strings for s. You must find the k-th in lexicographical order optimal escaped string or detect that the given k is too great.

Input

The first line of input contains the string s, consisting of pluses and hyphens. It's guaranteed that s is not empty and has a length of no more than 60 characters.

The second line contains a single integer k ($1 \le k \le 10^{18}$).

Output

You need to print the k-th in lexicographical order optimal escaped string for the string s specified in the input.

If k is greater than the number of optimal escaped strings, your program should print "`Overflow`".

Examples

standard input	standard output
++-- 2	++-{}-
-+-+- 2	Overflow