Exam Preparation – Task 2

2. Decompression

A simple approach to compressing text is to find sequences of repeating symbols (e. g. **abbbbccd**, contains **bbbb** which is a sequence of the symbol **b** repeated **4** times) and replace them with their length followed by the repeated symbol. For example, if we have **abbbbccd**, this method of compression will give us **a4bccd**.

Note that we only actually compress a sequence if the combination of the length and the symbol is shorter than the sequence – the sequence **a** is shorter than writing **1a**, the sequence **cc** is the same length as writing **2c**, so there is no point in replacing them – we leave them as they are.

You are given the version of a text compressed as described above. Your task is to "decompress" the text into what the original looked like before the compression. That means that every letter preceded by a positive integer number should be printed that number of times, whereas any letter preceded by another letter should be printed "as is".

Input

On the only line of the standard input you will receive the text to be decompressed – a sequence of lowercase English letters (**a**-**z**) and numbers (containing the digits **0**-**9**) with no other symbols (no spaces, no punctuation, etc.).

Output

A single line, containing the decompressed text.

Restrictions

The input will be such that the decompressed (output) text will contain no more than **20000** symbols.

The total running time of your program should be no more than **0.1s**

The total memory allowed for use by your program is **16MB**

Example I/O

Example Input	Expected Output
3abcc4de	aaabccdddde
10h3e14loo	hhhhhhhhhheeellllllllllllloo
3ab3de	aaabddde
waaq3p	waaqppp