Decode and Decrypt

So, being a KO-NE (Key Observation – Notification Expert), you really don't like the idea, that half the company you work for has started using a new method of messaging. This new method encrypts and encodes (compresses) the messages. Encoding is all well and good – company's saving on Broadband and all that jazz – but it's the encryption part you really have a problem with. After all, your job is "observation" and you really can't be effective at that when you can't even read the damn thing.

Good thing is, being as good as you are, you found the source of the idea for the messaging system – some old article, describing a primitive but effective encoding and encryption algorithm. So much for security by obscurity. Here's the encryption and encoding algorithm description from the article:

- We are given a **message** and a **cypher**
    - The message is the text the user wants to transmit
    - The cypher is a string which is used to encrypt and decrypt the message
    - The encrypted message is called **cypherText**
- We define a function **Encrypt**, which takes a message and a cypher:
    - It iterates over the symbols in the message and the cypher
    - For each **pair of symbols**, it takes their **codes** (in the table below) and computes the **bitwise XOR** of the **symbol in the message** with the **symbol in the cypher**.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

-
    - The **resulting code** is **summed** with the **ASCII code of the character 'A'** (65), giving a new ASCII code
    - The **character corresponding to this new ASCII code** is the **encrypted representation** of the respective **character in the message**
    - If the cypher string is shorter than the message, using it symbols loops to the beginning of the cypher. E.g. for a message "**ABCDE**" and a cypher "**PQR**" we will have:
        - **'A' encrypted with 'P'** = 'P', **'B' encrypted with 'Q'** = 'R', **'C' encrypted with 'R'** = 'T', **'D' encrypted with 'P'** = 'M', **'E' encrypted with 'Q'** = 'U'
    - If the message string is shorter than the cypher, some of its symbols will be encoded several times, until all of the cypher symbols are used.
        - E.g. for a message "**ABC**" and a cypher "**PQRST**", we will have:
        - **'A' encrypted with 'P'** and **the result** ('P') **encrypted with 'S'** = '^' (ASCII 94),
        - **'B' encrypted with 'Q'** and **the result** ('R') **encrypted with 'T'** = 'C',
        - **'C' encrypted only with 'R'** = 'T'
- We define a function **Encode**, which takes a string of text to compress:
    - It looks for sequences of symbols which are the same(e.g. 'aaaaa')

  - For each sequence of same symbols, the function replaces the sequence with a number representing the count of repeated symbols, followed immediately by the symbol which is repeated. This is called run-length encoding. E.g. **for the text "aaaabbbccccaa" we will have "4a3b4caa"**.
    - The function **replaces symbols** in the aforementioned way **ONLY if the run-length encoding** of the **same-symbol sequence** is **shorter than the sequence itself**
    - That's why in the example above the last two a's remain the same – '2a' has the same length as 'aa'
- Given the two functions, and given a message and a cypher, the encrypted message should be:
  - **Encode(Encrypt(message, cypher) + cypher) + lengthOfCypher**
    - '+' denotes concatenation, the two functions return strings and 'lengthOfCypher' is a number, which is equal to the number of symbols in the cypher
    - i.e. the message is encrypted with the cypher, then the cipher is added for decrypting purposes, then the result is compressed and a number denoting the length of the cypher before compression is added to the compressed string

Now, since you are very good, you know that the described **Encrypt function actually works both ways** – i.e. if something was encrypted with the function and a cypher, calling Encrypt again with the same cypher, but with the encrypted text, you will receive the original text.

- For example, **Encrypt("ABCDE", "PQR") = "PRTMU"** and **Encrypt("PRTMU", "PQR") = "ABCDE"**
  - Where the fist parameter of Encrypt is the message and the second is the cypher

The Encode function is also relatively easy to reverse – just take the numbers and print the symbol after each number the corresponding … number … of times.

Now all you have to do is put the pieces together and you can once again spy on the messaging in the company.

Write a program, which by **given an encrypted** (with the above described method) **message**, **recovers the original message**

Input

The input data should be read from the console.

On the only input line there will be the cyphered message

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data should be printed on the console. Print exactly one line – the original message (decrypted and decoded)

Input example	Output example
BKOXHI\EQOGX[YSOFTWARE8	TELERIKACADEMY
ABBAA6BA7	AAABB
KKICXDE3P5	JOHNY

Constraints

- All symbols in the message will have ASCII codes in the range [65; 127]
- The original message for any encrypted message will always contain only capital English letters
- The original message will be no longer than 1500 symbols
- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.