Some message encoding schemes require that an encoded message be sent in two parts. The first part, called the header, contains the characters of the message. The second part contains a pattern that represents the message. You must write a program that can decode messages under such a scheme.

The heart of the encoding scheme for your program is a sequence of "key" strings of 0's and 1's as follows:

$$0, 00, 01, 10, 000, 001, 010, 011, 100, 101, 110, 0000, 0001, \ldots, 1011, 1110, 00000, \ldots$$

The first key in the sequence is of length 1, the next 3 are of length 2, the next 7 of length 3, the next 15 of length 4, etc. If two adjacent keys have the same length, the second can be obtained from the first by adding 1 (base 2). Notice that there are no keys in the sequence that consist only of 1's.

The keys are mapped to the characters in the header in order. That is, the first key (0) is mapped to the first character in the header, the second key (00) to the second character in the header, the kth key is mapped to the kth character in the header. For example, suppose the header is:

AB#TANCnrtXc

Then 0 is mapped to A, 00 to B, 01 to #, 10 to T, 000 to A, ..., 110 to X, and 0000 to c.

The encoded message contains only 0's and 1's and possibly carriage returns, which are to be ignored. The message is divided into segments. The first 3 digits of a segment give the binary representation of the length of the keys in the segment. For example, if the first 3 digits are 010, then the remainder of the segment consists of keys of length 2 (00, 01, or 10). The end of the segment is a string of 1's which is the same length as the length of the keys in the segment. So a segment of keys of length 2 is terminated by 11. The entire encoded message is terminated by 000 (which would signify a segment in which the keys have length 0). The message is decoded by translating the keys in the segments one-at-a-time into the header characters to which they have been mapped.

Input

The input file contains several data sets. Each data set consists of a header, which is on a single line by itself, and a message, which may extend over several lines. The length of the header is limited only by the fact that key strings have a maximum length of 7 (111 in binary). If there are multiple copies of a character in a header, then several keys will map to that character. The encoded message contains only `0`'s and `1`'s, and it is a legitimate encoding according to the described scheme. That is, the message segments begin with the 3-digit length sequence and end with the appropriate sequence of `1`'s. The keys in any given segment are all of the same length, and they all correspond to characters in the header. The message is terminated by `000`.

Carriage returns may appear anywhere within the message part. They are *not* to be considered as part of the message.

Output

For each data set, your program must write its decoded message on a separate line. There should not be blank lines between messages.

Sample input

```
TNM AEIOU
0010101100011
1010001001110110011
11000
$#**\
0100000101101100011100101000
```

Sample output

```
TAN ME
##*\$
```