2005 Message decoding

You are to implement a message decoder for a scheme which sends messages in two parts: a header and the encoded message itself.

The foundation of the scheme is a sequence of binary 'key' strings as follows:

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

The first one is of length 1, the next three of length 2, the next seven of length 3, next fifteen of length 4, etc. If two adjacent keys have the same length, the second can be obtained from the first by adding 1. Note that there are no keys consisting only of 1's.

The keys are mapped to the characters in the header in order. Suppose the header is "`Bruce`", the first key (0) is mapped to '`B`', etc.

The encoded message contains only 0's and 1's and possibly carriage returns (new-lines) — though they 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 that segment. For example, if the first 3 digits are 011_2, then the remainder of the segment consists of keys of length 3_{10} (one of $000, 001, \ldots, 110$).

The end of a segment is a string of 1's which is the same length as the length of the keys in that segment (this is why all 1s were not used above). So, in the example above, the segment would be terminated by 111. The entire message is terminated by a segment begining 000 (which would signify a segment in which keys have length 0). The message is decoded by translating the keys in the segments into the header characters to which they have been mapped.

Input

The input file contains several datasets. The first line of each dataset contains the header, terminated by a new-line (which is not part of the message); any spaces are part of the header. The length of the header is limited by the fact that key strings have a maximum length of 111_2 (7_{10} in decimal). If there are multiple copies of a character in the header, then several keys will map to that character.

The remaining lines of input contain the encoded message. The message is made up only of '`0`'s and '`1`'s (and possibly carriage-returns — to be ignored). You are guaranteed that it is a valid message according to the encoding described above.

Output

For each dataset, your output must consist of the message, followed by a new-line.

Sample Input

```
$#**/
01
00000
1011011
000111
00101000
```

Sample Output

```
##*/$
```