K Kracked Enkoder

Time limit: 1s

Encoding and decoding play important roles in many computer systems. Especially for Karl, whose computer can only store at most 29 bits! Not knowing how to deal with this, Karl has turned to you to find a way to encode a number so that it can be stored on his computer, and to decode the number as well.

Karl's Potato Computer.
Internet meme at imgflip, fair use

This is a *multi-pass* problem: your program will be invoked two times: one run for encoding, and one run for decoding.

In the first run, your program receives the string "`encode`", together with a positive integer $1 \leq x \leq 2^{30} - 2$. This is the number that Karl wants you to encode. Your program must then output a non-empty string, consisting of at most 29 zeros and ones.

In the second run, your program receives the string "`decode`", together with the output of the first run of your program. Your program must then output Karl's number x.

Input

The input consists of:

- One line with the action that your program needs to perform: either the string "`encode`" or "`decode`".
- If the action is "`encode`":
  - One line with a string number x ($1 \leq x \leq 2^{30} - 2$), the number Karl wants you to encode.
- If the action is "`decode`":
  - One line with a binary string b ($1 \leq |b| \leq 29$), only consisting of digits 0 and 1. This is the binary string you used to encode Karl's number x.

This is a multi-pass problem. For each test case, your program will be invoked two times. It is guaranteed that the first pass is a "`encode`" action, and that the second pass is a "`decode`" action.

This problem does *not* provide a testing tool.

Output

If the action is "`encode`", output a string b of length $1 \leq |b| \leq 29$, only consisting of digits 0 and 1.

If the action is "`decode`", output the original number x.

Sample Case 1

Input	**Pass 1**	**Output**
`encode` `123456`	`100110011001`	

Input	**Pass 2**	**Output**
`decode` `100110011001`	`123456`	