esreveR Order

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

This is a run-twice problem.

You have an array of $n \leq 1000$ unsigned 64-bit integers. You want to quickly transmit it to another computer. To do this, you send numbers through the Internet in parallel and then reassemble the array.

However, an unexpected problem has arisen. As is known, the "network" byte order in a multi-byte number differs from the order used in modern computers. Specifically, in a modern computer, bytes are written from least significant to most significant (little-endian), and in network byte order, bytes are written from most significant to least significant (big-endian). During the conversion, each number is written as a sequence of 8 bytes, and the bytes are written in reverse order. And in some cases, due to server failures, the reverse conversion was not performed...

So, you send an array, and then receive another array where each element can arrive either in the usual little-endian order or in the network big-endian order. The order of the array elements is preserved. When transmitting, you can use no more than 1024 64-bit integers (in other words, no more than 8 **kibibytes**). Your task is to restore the original array after possible changes.

Input

If you need to transmit an array, the first line contains the word "`encode`", the second line contains an integer n ($1 \leq n \leq 1000$), and the third line contains n integers in the range from 0 to $2^{64} - 1$.

If you need to receive an array, the first line contains the word "`decode`", the second line contains an integer k ($k \leq 1024$), and the third line contains k integers in the range from 0 to $2^{64} - 1$. It is guaranteed that the numbers are in the same order as they were transmitted, and that each number is either transmitted unchanged or has its byte order reversed (according to the problem statement).

Output

In the case of transmitting an array, output one integer $k \leq 1024$ on the first line: the number of integers to transmit. On the second line, output k integers in the range from 0 to $2^{64} - 1$.

In the case of receiving an array, output n integers on a single line: the original array.

Examples

standard input	*standard output*
encode 3 15 10 2023	6 15 15 10 10 2023 2023
decode 6 15 15 10 720575940379279360 1664727454759832780 8 2023	15 10 2023

Note

In the lower example, in the case of receiving an array, all six numbers will be given on a single line. An additional line break is added for readability.

On each test, your program will be run twice: first for transmitting the array, and then for receiving it. The output of the first run with possible changes will be the input of the second run. Your solution passes a test if the original array is restored correctly.