Problem H. Hash Server

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

This is an interactive, run-twice problem.

There is a hash server whose purpose is to hash strings of length 10. However, it has a problem: it is about to stop working, as it can only handle 100 more strings. Your task is to write a program that can replicate the server's behavior.

A string s is hashed as follows. Let s_i be the 1-based alphabetic number of the i-th letter. The hash is calculated using the following formula:

$$\sum_{i=1}^{10}(s_i \cdot x^{k_i}) \cdot (a_i + s_i) \pmod{26^{10}},$$

where the parameters a_i, k_i ($1 \le i \le 10$), and x ($x > \max_{i=1}^{10}(\max(a_i, k_i))$) are **distinct prime** numbers from 1 to 10^9, set on the hash server. These parameters are unknown constants.

To generate a response, the hash server converts the number into a string of length 10. It is written in the positional numeral system with base 26, but every digit is represented by a letter: a represents digit 0, b represents digit 1, etc. If the resulting number is too short, leading zeroes are added to make it exactly 10 digits long.

Your program will run twice. During the first run, your program can make at most 100 unique requests to the server. For every request, the hash server returns the hash of the given string.

During the second run, your program will receive a list of server responses from the first run in an arbitrary order. Your program must then process 100 hash requests from the jury's program and produce the same responses as the original server.

Interaction Protocol

At the beginning, the participant's program receives a line with a single integer: either 1 or 2, indicating which run is being executed.

If this is the first run, the participant's program should output one string containing exactly 10 lowercase English letters. In response, the jury's program will output a string in the same format representing the hash of the requested string. All requests **must be different**. After the last request, the participant's program should output the string "done" on a separate line.

If this is the second run, the jury's program will first output one integer k indicating the number of responses from the first run, and then output a list of k responses in arbitrary order, each on a separate line. After that, the jury's program will make hash requests to the participant's program by outputting strings in the same format as in the first run. For every request, the participant's program should output a 10-letter string which will be considered as the answer to the problem: the correct hash for the given request. The jury's program makes exactly 100 requests. The list of requests and their order are fixed for each specific test and do not adapt to the participant's program output.

Example

standard input	standard output
1 wilyevxwyy gcfffvmuie nykaeyifai omhdftcmgu wqjmrukrfi	 aaaaaaaaaa bbbbbbbbbb abababababa bababababa aaaaabbbbb done
2 5 nykaeyifai omhdftcmgu gcfffvmuie wilyevxwyy wqjmrukrfi aaaaaaaaaa bbbbbbbbbb cccccccccc dddddddddd eeeeeeeeee ffffffffff gggggggggg ...	 wilyevxwyy gcfffvmuie dhfvcyssbs nxntwfpqfo lzdblqdots xlzrxeinse wkdrewenay ...

Explanation

The example of the second run contains only 7 requests for brevity. In the testing system, the jury's program will make 100 requests in the first test.

Here are the parameters that are used in the sample:

- $x = 73$.
- $a = (71, 67, 61, 59, 53, 47, 43, 41, 37, 31)$.
- $k = (29, 23, 19, 17, 13, 11, 7, 5, 3, 2)$.