F Faulty Connection

Time limit: 1s

It is the 25th of June, 1825. English inventer William Fothergill Cooke and you, English scientist Charles Wheatstone, are working on an early prototype of an electrical telegraph system.[3] This system consists of a transmitter and a receiver. The transmitter can send messages to the receiver, which we model as a sequence of positive integers. However, the prototype is far from flawless, so quite often, an integer that is transmitted is not actually received. Integers never get altered through the connection, though, so all received integers are guaranteed to have been transmitted.



The Cooke and Wheatstone electric telegraph can transmit letters with the use of 5 needles which point to the transmitted letters. CC BY-SA 4.0 by Geni on Wikimedia Commons

To remedy this flaw, Cooke and you are working on an error correction code. The idea is to agree on a fixed list of 600 possible messages, indexed 1 to 600, and assign each message a list of 30 positive integers of at most 1000. To send a message from the list, you simply send the corresponding list of integers. Sometimes, some integers may get lost or arrive in a different order, but if enough integers remain, the hope is that there is still only one possible message.

When trying this out back in April, you noticed that, with some bad luck, the connection can get very faulty. Sometimes only two integers remained, which can easily make a message ambiguous! You decide to investigate whether this issue can be resolved purely by improving the error correction code. To each of the 600 possible messages, you need to assign a list of 30 positive integers of at most 1000, such that receiving any two integers in any order from any one of the lists uniquely determines the corresponding message.

Your program will be run multiple times for each test case. In the first pass, your program will be given an index of a message to send, which your program should assign a list of 30 integers. In subsequent passes, your program will be given two of the 30 integers from the first pass in any order, which it should then decode to retrieve the original message.

Your submission may take up to 1 second for each pass.

A testing tool is provided to help you develop your solution.

[3] One may point out that Cooke and Wheatstone actually did not release a telegraph design until 1837, but obviously, this is due to the problem we are trying to solve here! :)

Input

This is a *multi-pass* problem.

The input consists of:

- One line with the action your program needs to perform:
  - "`send`" if you need to send a message.
  - "`receive`" if you need to receive two integers and determine the corresponding message.
- If the action is "`send`", one line with an integer k ($1 \leq k \leq 600$), the index of the message to send.
- If the action is "`receive`", one line with two distinct integers a and b ($1 \leq a, b \leq 1000$) from the output of your program in the first pass, in any order.

Output

If the action is "`send`", output 30 distinct integers x ($1 \leq x \leq 1000$), assigning this list of integers to the message with index k.

If the action is "`receive`", output the index k of the corresponding message.

Note

The displayed sample output of a "`send`" action uses line wrapping for display purposes. Whitespace in the output is treated as usual.

Sample Case 1

Input	**Pass 1**	**Output**

`send`
`42`
	`814 734 58 792 286 974 893 735`
	`538 498 916 163 226 32 160 659`
	`980 994 775 334 44 492 276 983`
	`398 885 179 888 755 121`	

Input	**Pass 2**	**Output**
`receive` `286 58`	`42`	

Input	**Pass 3**	**Output**
`receive` `163 492`	`42`	