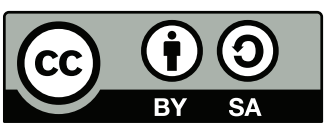
Freshmen Programming Contest
Test Session

Problems

X	Alternative Encryption
Y	Floating Fowl
Z	Test Case Troubles



X Alternative Encryption

Time limit: 1s

In the fight against theft of intellectual property by rivalling universities, TU Delft has decided to implement a system of secure communication, to be called New Well-Encrypted Remote Communication. All internal communication will be encrypted before being sent over the network, and then decrypted upon arrival. They have already set up the infrastructure, they have come to you for the encryption.

The enigma, an encryption machine that solves this problem. CC BY 2.0 by William Warby on Wikimedia Commons

You are tasked to design an algorithm for both encryption and decryption of text consisting of only English lowercase letters. You do not need to tell them how you do it, in fact, they encourage you to keep it a secret, to improve security. Your algorithm only needs to meet the following criteria:

- No matter what the text is, encrypting it and then decrypting the result should obviously yield the original text.
- To make sure there are no issues with the transmission, the encrypted text should also consist of only English lowercase letters and should have the same number of letters as the original text.
- To make sure the code cannot be broken easily, for all i, the ith letter of the encrypted text should differ from the ith letter of the original text.

The word "`nwercjury`" for example may not be encrypted as "`irritating`", because the number of letters does not match. Nor may you encrypt it as "`imbecilic`", since the fifth letter is a '`c`' in both. An example of an acceptable encryption is "`fantastic`" (both have an '`n`' and a '`c`', but in different positions).

Your program will be run twice for each test case. In the first pass, your program will be given a number of strings to encrypt. In the second pass, your program will be given the strings as encrypted by the first pass, which it should then decrypt to retrieve the original input. Your submission may take up to 1 second for each pass.

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

Input

The input consists of:

- One line with either "`encrypt`" or "`decrypt`", indicating the action your program has to perform.
- One line with an integer n ($1 \leq n \leq 1000$), the number of strings.
- n lines, each with a string s ($1 \leq |s| \leq 100$), the text to encrypt or decrypt.
  All input strings consist of only English lowercase letters (`a`–`z`).

Output

For each string, output its encryption or decryption, as required.

Sample Case 1

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

`encrypt`
`3`
`plaintext`
`nwerc`
`correct`
	`encrypted`
	`delft`
	`balloon`	

Input	**Pass 2**	**Output**

`decrypt`
`3`
`encrypted`
`delft`
`balloon`
	`plaintext`
	`nwerc`
	`correct`	

Y Floating Fowl

Time limit: 1s

You and your friend have both built enormous rubber ducks. Now the two of you are fighting: you think yours will float better, while your friend thinks theirs will float better. Of course, the duck that will float better is simply the one with the lowest density, that is, the least weight per volume.

Given the weights and volumes of the two ducks, can you determine which one will float better? And how much better does it float?

Big rubber ducks.
CC BY-SA 4.0 by LN9267 on
Wikimedia Commons, cropped

Input

The input consists of:

- One line with two integers w_1 and v_1 ($1 \leq w_1, v_1 \leq 10^{18}$), the weight and volume of the first rubber duck.
- One line with two integers w_2 and v_2 ($1 \leq w_2, v_2 \leq 10^{18}$), the weight and volume of the second rubber duck.

Output

Provide two outputs:

- Output "`first`" if the first duck floats better, "`second`" if the second duck floats better, or "`equal`" if they will float equally well.
- A number that indicates how much better the less dense duck floats: the difference between the density of the denser duck and the density of the less dense duck.

  Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1

```
5 3
4 3
```

Sample Output 1

```
second
0.333333333
```

Sample Input 2

```
1 8
1 7
```

Sample Output 2

```
first
0.01785714285714285
```

Sample Input 3

```
10 5
4 2
```

Sample Output 3

```
equal
0
```

This page is intentionally left blank.

Z Test Case Troubles

Time limit: 3s

A lonely jury member in its natural habitat, making test cases in the night time

The typical jury member for the FPC is a hard-working algorithms enthusiast. Approaching the main contest, however, the jury had some trouble finishing all test cases in time. Luckily, you can help them out by making some tricky and clever test cases!

For one problem, test cases have been made. but the jury is afraid that the worst-case is not properly tested and asks you to spice up the current test cases. The test cases consist of a list of integers. Your job is to make sure that the integers are neither in ascending nor in descending[1] order.

Input

- The first line contains an integer n ($0 \leq n \leq 10^5$), the number of integers.
- Then n lines follow, where each line contains one integer k ($0 \leq k \leq 10^9$).

Output

Print the integers, one on each line, from the input in a non-ascending and non-descending order. If no such order exists, print "`impossible`".

Sample Input 1	Sample Output 1
3 1 2 3	2 1 3

Sample Input 2	Sample Output 2
3 37 37 37	impossible

Sample Input 3	Sample Output 3
5 9 8 7 6 5	6 5 7 8 9

[1] A list $x_1, \ldots, x_n$ is *ascending* when $x_i \leq x_{i+1}$ for all $1 \leq i < n$, and *descending* when $x_i \geq x_{i+1}$ for all $1 \leq i < n$.