A. Voting for Photos

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

After celebrating the midcourse the students of one of the faculties of the Berland State University decided to conduct a vote for the best photo. They published the photos in the social network and agreed on the rules to choose a winner: the photo which gets most likes wins. If multiple photoes get most likes, the winner is the photo that gets this number first.

Help guys determine the winner photo by the records of likes.

Input

The first line of the input contains a single integer n ($1 \le n \le 1000$) — the total likes to the published photoes.

The second line contains n positive integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 1\,000\,000$), where a_i is the identifier of the photo which got the i-th like.

Output

Print the identifier of the photo which won the elections.

Examples

input

```
5
1 3 2 2 1
```

output

```
2
```

input

```
9
100 200 300 200 100 300 300 100 200
```

output

```
300
```

Note

In the first test sample the photo with id 1 got two likes (first and fifth), photo with id 2 got two likes (third and fourth), and photo with id 3 got one like (second).

Thus, the winner is the photo with identifier 2, as it got:

- more likes than the photo with id 3;
- as many likes as the photo with id 1, but the photo with the identifier 2 got its second like earlier.

B. Chat Order

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp is a big lover of killing time in social networks. A page with a chatlist in his favourite network is made so that when a message is sent to some friend, his friend's chat rises to the very top of the page. The relative order of the other chats doesn't change. If there was no chat with this friend before, then a new chat is simply inserted to the top of the list.

Assuming that the chat list is initially empty, given the sequence of Polycaprus' messages make a list of chats after all of his messages are processed. Assume that no friend wrote any message to Polycarpus.

Input

The first line contains integer n ($1 \le n \le 200\,000$) — the number of Polycarpus' messages. Next n lines enlist the message recipients in the order in which the messages were sent. The name of each participant is a non-empty sequence of lowercase English letters of length at most 10.

Output

Print all the recipients to who Polycarp talked to in the order of chats with them, from top to bottom.

Examples

input

```
4
alex
ivan
roman
ivan
```

output

```
ivan
roman
alex
```

input

```
8
alina
maria
ekaterina
darya
darya
ekaterina
maria
alina
```

output

```
alina
maria
ekaterina
darya
```

Note

In the first test case Polycarpus first writes to friend by name "`alex`", and the list looks as follows:

1. `alex`

Then Polycarpus writes to friend by name "`ivan`" and the list looks as follows:

1. `ivan`
2. `alex`

Polycarpus writes the third message to friend by name "`roman`" and the list looks as follows:

1. `roman`
2. `ivan`
3. `alex`

Polycarpus writes the fourth message to friend by name "`ivan`", to who he has already sent a message, so the list of chats changes as follows:

1. `ivan`
2. `roman`
3. `alex`

C. Promocodes with Mistakes

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

During a New Year special offer the "Sudislavl Bars" offered n promo codes. Each promo code consists of exactly six digits and gives right to one free cocktail at the bar "Mosquito Shelter". Of course, all the promocodes differ.

As the "Mosquito Shelter" opens only at 9, and partying in Sudislavl usually begins at as early as 6, many problems may arise as to how to type a promotional code without errors. It is necessary to calculate such maximum k, that the promotional code could be uniquely identified if it was typed with no more than k errors. At that, $k = 0$ means that the promotional codes must be entered exactly.

A mistake in this problem should be considered as entering the wrong numbers. For example, value "`123465`" contains two errors relative to promocode "`123456`". Regardless of the number of errors the entered value consists of exactly six digits.

Input

The first line of the output contains number n ($1 \leq n \leq 1000$) — the number of promocodes.

Each of the next n lines contains a single promocode, consisting of exactly 6 digits. It is guaranteed that all the promocodes are distinct. Promocodes can start from digit "`0`".

Output

Print the maximum k (naturally, not exceeding the length of the promocode), such that any promocode can be uniquely identified if it is typed with at most k mistakes.

Examples

input

```
2
000000
999999
```

output

```
2
```

input

```
6
211111
212111
222111
111111
112111
121111
```

output

```
0
```

Note

In the first sample $k < 3$, so if a bar customer types in value "`090909`", then it will be impossible to define which promocode exactly corresponds to it.

D. Running with Obstacles

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A sportsman starts from point $x_{start} = 0$ and runs to point with coordinate $x_{finish} = m$ (on a straight line). Also, the sportsman can jump — to jump, he should first take a run of length of not less than s meters (in this case for these s meters his path should have no obstacles), and after that he can jump over a length of not more than d meters. Running and jumping is permitted only in the direction from left to right. He can **start** and**finish** a jump only at the points with integer coordinates in which there are **no obstacles**. To overcome some obstacle, it is necessary to land at a point which is strictly to the right of this obstacle.

On the way of an athlete are n obstacles at coordinates $x_1, x_2, \ldots, x_n$. He cannot go over the obstacles, he can only jump over them. Your task is to determine whether the athlete will be able to get to the finish point.

Input

The first line of the input containsd four integers n, m, s and d ($1 \le n \le 200\,000$, $2 \le m \le 10^9$, $1 \le s, d \le 10^9$) — the number of obstacles on the runner's way, the coordinate of the finishing point, the length of running before the jump and the maximum length of the jump, correspondingly.

The second line contains a sequence of n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le m - 1$) — the coordinates of the obstacles. It is guaranteed that the starting and finishing point have no obstacles, also no point can have more than one obstacle, The coordinates of the obstacles are given in an arbitrary order.

Output

If the runner cannot reach the finishing point, print in the first line of the output "`IMPOSSIBLE`" (without the quotes).

If the athlete can get from start to finish, print any way to do this in the following format:

- print a line of form "`RUN X>`" (where "X" should be a positive integer), if the athlete should run for "X" more meters;
- print a line of form "`JUMP Y`" (where "Y" should be a positive integer), if the sportsman starts a jump and should remain in air for "Y" more meters.

All commands "`RUN`" and "`JUMP`" should strictly alternate, starting with "`RUN`", besides, they should be printed chronologically. It is not allowed to jump over the finishing point but it is allowed to land there after a jump. The athlete should stop as soon as he reaches finish.

Examples

input

```
3 10 1 3
3 4 7
```

output

```
RUN 2
JUMP 3
RUN 1
JUMP 2
RUN 2
```

input

```
2 9 2 3
6 4
```

output

```
IMPOSSIBLE
```