CROC-MBTU 2012, Final Round

A. Paper Work

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

Polycarpus has been working in the analytic department of the "F.R.A.U.D." company for as much as n days. Right now his task is to make a series of reports about the company's performance for the last n days. We know that the main information in a day report is value a_i, the company's profit on the i-th day. If a_i is negative, then the company suffered losses on the i-th day.

Polycarpus should sort the daily reports into folders. Each folder should include data on the company's performance for several consecutive days. Of course, the information on each of the n days should be exactly in one folder. Thus, Polycarpus puts information on the first few days in the first folder. The information on the several following days goes to the second folder, and so on.

It is known that the boss reads one daily report folder per day. If one folder has three or more reports for the days in which the company suffered losses ($a_i < 0$), he loses his temper and his wrath is terrible.

Therefore, Polycarpus wants to prepare the folders so that none of them contains information on three or more days with the loss, and the number of folders is minimal.

Write a program that, given sequence a_i, will print the minimum number of folders.

Input

The first line contains integer n ($1 \le n \le 100$), n is the number of days. The second line contains a sequence of integers $a_1, a_2, ..., a_n$ ($|a_i| \le 100$), where a_i means the company profit on the i-th day. It is possible that the company has no days with the negative a_i.

Output

Print an integer k — the required minimum number of folders. In the second line print a sequence of integers $b_1, b_2, ..., b_k$, where b_j is the number of day reports in the j-th folder.

If there are multiple ways to sort the reports into k days, print any of them.

Sample test(s)

input
11 1 2 3 -4 -5 -6 5 -5 -6 -7 6
output
3 5 3 3

input
5 0 -1 100 -1 0
output
1 5

Note

Here goes a way to sort the reports from the first sample into three folders:

```
1 2 3 -4 -5|-6 5 -5|-6 -7 6
```

In the second sample you can put all five reports in one folder.

B. Restoring IPv6

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

An IPv6-address is a 128-bit number. For convenience, this number is recorded in blocks of 16 bits in hexadecimal record, the blocks are separated by colons — 8 blocks in total, each block has four hexadecimal digits. Here is an example of the correct record of a IPv6 address: "0124:5678:90ab:cdef:0124:5678:90ab:cdef". We'll call such format of recording an IPv6-address *full*.

Besides the full record of an IPv6 address there is a *short* record format. The record of an IPv6 address can be shortened by removing one or more leading zeroes at the beginning of each block. However, each block should contain at least one digit in the short format. For example, the leading zeroes can be removed like that: "a56f:00d3:0000:0124:0001:f19a:1000:0000" → "a56f:d3:0:0124:01:f19a:1000:00". There are more ways to shorten zeroes in this IPv6 address.

Some IPv6 addresses contain long sequences of zeroes. Continuous sequences of 16-bit zero blocks can be shortened to "::". A sequence can consist of one or several **consecutive blocks**, with all 16 bits equal to 0.

You can see examples of zero block shortenings below:

- "a56f:00d3:0000:0124:0001:0000:0000:0000" → "a56f:00d3:0000:0124:0001::";
- "a56f:0000:0000:0124:0001:0000:1234:0ff0" → "a56f::0124:0001:0000:1234:0ff0";
- "a56f:0000:0000:0000:0001:0000:1234:0ff0" → "a56f:0000::0000:0001:0000:1234:0ff0";
- "a56f:00d3:0000:0124:0001:0000:0000:0000" → "a56f:00d3:0000:0124:0001::0000";
- "0000:0000:0000:0000:0000:0000:0000:0000" → "::".

It is not allowed to shorten zero blocks in the address more than once. This means that the short record can't contain the sequence of characters "::" more than once. Otherwise, it will sometimes be impossible to determine the number of zero blocks, each represented by a double colon.

The format of the record of the IPv6 address after removing the leading zeroes and shortening the zero blocks is called *short*.

You've got several short records of IPv6 addresses. Restore their full record.

Input

The first line contains a single integer n — the number of records to restore ($1 \le n \le 100$).

Each of the following n lines contains a string — the short IPv6 addresses. Each string only consists of string characters "0123456789abcdef:".

It is guaranteed that each short address is obtained by the way that is described in the statement from some full IPv6 address.

Output

For each short IPv6 address from the input print its full record on a separate line. Print the full records for the short IPv6 addresses in the order, in which the short records follow in the input.

Sample test(s)

input
6 a56f:d3:0:0124:01:f19a:1000:00 a56f:00d3:0000:0124:0001:: a56f::0124:0001:0000:1234:0ff0 a56f:0000::0000:0001:0000:1234:0ff0 :: 0ea::4d:f4:6:0
output
a56f:00d3:0000:0124:0001:f19a:1000:0000 a56f:00d3:0000:0124:0001:0000:0000:0000 a56f:0000:0000:0124:0001:0000:1234:0ff0 a56f:0000:0000:0000:0001:0000:1234:0ff0 0000:0000:0000:0000:0000:0000:0000:0000 00ea:0000:0000:0000:004d:00f4:0006:0000

C. Movie Critics

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

A film festival is coming up in the city N. The festival will last for exactly n days and each day will have a premiere of exactly one film. Each film has a genre — an integer from 1 to k.

On the i-th day the festival will show a movie of genre a_i. We know that a movie of each of k genres occurs in the festival programme at least once. In other words, each integer from 1 to k occurs in the sequence $a_1, a_2, ..., a_n$ at least once.

Valentine is a movie critic. He wants to watch some movies of the festival and then describe his impressions on his site.

As any creative person, Valentine is very susceptive. After he watched the movie of a certain genre, Valentine forms the *mood* he preserves until he watches the next movie. If the genre of the next movie is the same, it does not change Valentine's mood. If the genres are different, Valentine's mood changes according to the new genre and Valentine has a *stress*.

Valentine can't watch all n movies, so he decided to exclude from his to-watch list movies of one of the genres. In other words, Valentine is going to choose exactly one of the k genres and will skip all the movies of this genre. He is sure to visit other movies.

Valentine wants to choose such genre x ($1 \le x \le k$), that the total number of after-movie stresses (after all movies of genre x are excluded) were minimum.

Input

The first line of the input contains two integers n and k ($2 \le k \le n \le 10^5$), where n is the number of movies and k is the number of genres.

The second line of the input contains a sequence of n positive integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le k$), where a_i is the genre of the i-th movie. It is guaranteed that each number from 1 to k occurs at least once in this sequence.

Output

Print a single number — the number of the genre (from 1 to k) of the excluded films. If there are multiple answers, print the genre with the minimum number.

Sample test(s)

input
10 3 1 1 2 3 2 3 3 1 1 3
output
3

input
7 3 3 1 3 2 3 1 2
output
1

Note

In the first sample if we exclude the movies of the 1st genre, the genres $2, 3, 2, 3, 3, 3$ remain, that is 3 stresses; if we exclude the movies of the 2nd genre, the genres $1, 1, 3, 3, 3, 1, 1, 3$ remain, that is 3 stresses; if we exclude the movies of the 3rd genre the genres $1, 1, 2, 2, 1, 1$ remain, that is 2 stresses.

In the second sample whatever genre Valentine excludes, he will have exactly 3 stresses.

D. Building Bridge

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

Two villages are separated by a river that flows from the north to the south. The villagers want to build a bridge across the river to make it easier to move across the villages.

The river banks can be assumed to be vertical straight lines $x = a$ and $x = b$ ($0 < a < b$).

The west village lies in a steppe at point $O = (0, 0)$. There are n pathways leading from the village to the river, they end at points $A_i = (a, y_i)$. The villagers there are plain and simple, so their pathways are straight segments as well.

The east village has reserved and cunning people. Their village is in the forest on the east bank of the river, but its exact position is not clear. There are m twisted paths leading from this village to the river and ending at points $B_i = (b, y'_i)$. The lengths of all these paths are known, the length of the path that leads from the eastern village to point B_i, equals l_i.

The villagers want to choose exactly one point on the left bank of river A_i, exactly one point on the right bank B_j and connect them by a straight-line bridge so as to make the total distance between the cities (the sum of $|OA_i| + |A_iB_j| + l_j$, where $|XY|$ is the Euclidean distance between points X and Y) were minimum. The Euclidean distance between points (x_1, y_1) and (x_2, y_2) equals $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Help them and find the required pair of points.

Input

The first line contains integers n, m, a, b ($1 \le n, m \le 10^5$, $0 < a < b < 10^6$).

The second line contains n integers in the ascending order: the i-th integer determines the coordinate of point A_i and equals y_i ($|y_i| \le 10^6$).

The third line contains m integers in the ascending order: the i-th integer determines the coordinate of point B_i and equals y'_i ($|y'_i| \le 10^6$).

The fourth line contains m more integers: the i-th of them determines the length of the path that connects the eastern village and point B_i, and equals l_i ($1 \le l_i \le 10^6$).

It is guaranteed, that there is such a point C with abscissa at least b, that $|B_iC| \le l_i$ for all i ($1 \le i \le m$). It is guaranteed that no two points A_i coincide. It is guaranteed that no two points B_i coincide.

Output

Print two integers — the numbers of points on the left (west) and right (east) banks, respectively, between which you need to build a bridge. You can assume that the points on the west bank are numbered from 1 to n, in the order in which they are given in the input. Similarly, the points on the east bank are numbered from 1 to m in the order in which they are given in the input.

If there are multiple solutions, print any of them. The solution will be accepted if the final length of the path will differ from the answer of the jury by no more than 10^{-6} in absolute or relative value.

Sample test(s)

input
3 2 3 5 -2 -1 4 -1 2 7 3
output
2 2

E. Mad Joe

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

Joe has been hurt on the Internet. Now he is storming around the house, destroying everything in his path.

Joe's house has n floors, each floor is a segment of m cells. Each cell either contains nothing (it is an empty cell), or has a brick or a concrete wall (always something one of three). It is believed that each floor is surrounded by a concrete wall on the left and on the right.

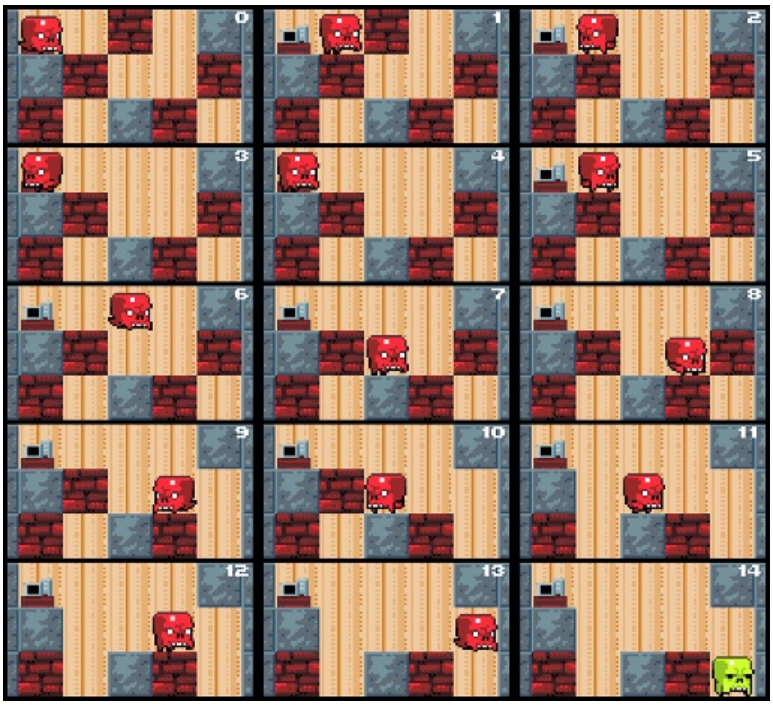
Now Joe is on the n-th floor and in the first cell, counting from left to right. At each moment of time, Joe has the direction of his gaze, to the right or to the left (always one direction of the two). Initially, Joe looks to the right.

Joe moves by a particular algorithm. Every second he makes one of the following actions:

- If the cell directly under Joe is empty, then Joe falls down. That is, he moves to this cell, the gaze direction is preserved.
- Otherwise consider the next cell in the current direction of the gaze.
  - If the cell is empty, then Joe moves into it, the gaze direction is preserved.
  - If this cell has bricks, then Joe breaks them with his forehead (the cell becomes empty), and changes the direction of his gaze to the opposite.
  - If this cell has a concrete wall, then Joe just changes the direction of his gaze to the opposite (concrete can withstand any number of forehead hits).

Joe calms down as soon as he reaches **any** cell of the first floor.

The figure below shows an example Joe's movements around the house.



Determine how many seconds Joe will need to calm down.

Input

The first line contains two integers n and m ($2 \le n \le 100$, $1 \le m \le 10^4$).

Next n lines contain the description of Joe's house. The i-th of these lines contains the description of the $(n - i + 1)$-th floor of the house — a line that consists of m characters: "`.`" means an empty cell, "`+`" means bricks and "`#`" means a concrete wall.

It is guaranteed that the first cell of the n-th floor is empty.

Output

Print a single number — the number of seconds Joe needs to reach the first floor; or else, print word "`Never`" (without the quotes), if it can never happen.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Sample test(s)

input
3 5 ..+.# #+..+

```
+.#+.
```

output

```
14
```

input

```
4 10
...+.##+.+
+#++..+++#
++.#++++..
.+##.++#.+
```

output

```
42
```

input

```
2 2
..
++
```

output

```
Never
```