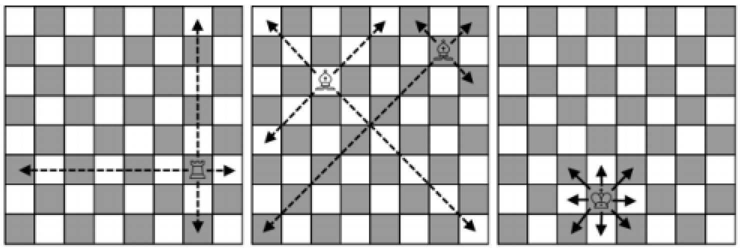
A. Rook, Bishop and King

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

Little Petya is learning to play chess. He has already learned how to move a king, a rook and a bishop. Let us remind you the rules of moving chess pieces. A chessboard is 64 square fields organized into an 8×8 table. A field is represented by a pair of integers (r, c) — the number of the row and the number of the column (in a classical game the columns are traditionally indexed by letters). Each chess piece takes up exactly one field. To make a move is to move a chess piece, the pieces move by the following rules:

- A rook moves any number of fields horizontally or vertically.
- A bishop moves any number of fields diagonally.
- A king moves one field in any direction — horizontally, vertically or diagonally.


The pieces move like that

Petya is thinking about the following problem: what minimum number of moves is needed for each of these pieces to move from field (r_1, c_1) to field (r_2, c_2)? At that, we assume that there are no more pieces besides this one on the board. Help him solve this problem.

Input

The input contains four integers r_1, c_1, r_2, c_2 $(1 \le r_1, c_1, r_2, c_2 \le 8)$ — the coordinates of the starting and the final field. The starting field doesn't coincide with the final one.

You can assume that the chessboard rows are numbered from top to bottom 1 through 8, and the columns are numbered from left to right 1 through 8.

Output

Print three space-separated integers: the minimum number of moves the rook, the bishop and the king (in this order) is needed to move from field (r_1, c_1) to field (r_2, c_2). If a piece cannot make such a move, print a 0 instead of the corresponding number.

Sample test(s)

input
```
4 3 1 6
```
output
```
2 1 3
```

input
```
5 5 5 6
```
output
```
1 0 1
```

B. Berland Bingo

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Lately, a national version of a bingo game has become very popular in Berland. There are n players playing the game, each player has a card with numbers. The numbers on each card are distinct, but distinct cards can have equal numbers. The card of the i-th player contains m_i numbers.

During the game the host takes numbered balls one by one from a bag. He reads the number aloud in a high and clear voice and then puts the ball away. All participants cross out the number if it occurs on their cards. The person who crosses out all numbers from his card first, wins. If multiple people cross out all numbers from their cards at the same time, there are no winners in the game. At the beginning of the game the bag contains 100 balls numbered 1 through 100, the numbers of all balls are distinct.

You are given the cards for each player. Write a program that determines whether a player can win the game at the most favorable for him scenario or not.

Input

The first line of the input contains integer n $(1 \le n \le 100)$ — the number of the players. Then follow n lines, each line describes a player's card. The line that describes a card starts from integer m_i $(1 \le m_i \le 100)$ that shows how many numbers the i-th player's card has. Then follows a sequence of integers $a_{i, 1}, a_{i, 2}, ..., a_{i, m_i}$ $(1 \le a_{i, k} \le 100)$ — the numbers on the i-th player's card. The numbers in the lines are separated by single spaces.

It is guaranteed that all the numbers on each card are distinct.

Output

Print n lines, the i-th line must contain word "`YES`" (without the quotes), if the i-th player can win, and "`NO`" (without the quotes) otherwise.

Sample test(s)

input

```
3
1 1
3 2 4 1
2 10 11
```

output

```
YES
NO
YES
```

input

```
2
1 1
1 1
```

output

```
NO
NO
```

C. Mittens

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

A Christmas party in city S. had n children. All children came in mittens. The mittens can be of different colors, but each child had the left and the right mitten of the same color. Let's say that the colors of the mittens are numbered with integers from 1 to m, and the children are numbered from 1 to n. Then the i-th child has both mittens of color c_i.

The Party had Santa Claus ('Father Frost' in Russian), his granddaughter Snow Girl, the children danced around the richly decorated Christmas tree. In fact, everything was so bright and diverse that the children wanted to wear mittens of distinct colors. The children decided to swap the mittens so that each of them got one left and one right mitten in the end, and these two mittens were of distinct colors. All mittens are of the same size and fit all the children.

The children started exchanging the mittens haphazardly, but they couldn't reach the situation when each child has a pair of mittens of distinct colors. Vasily Petrov, the dad of one of the children, noted that in the general case the children's idea may turn out impossible. Besides, he is a mathematician and he came up with such scheme of distributing mittens that the number of children that have distinct-colored mittens was maximum. You task is to repeat his discovery. Note that the left and right mittens are different: each child must end up with one left and one right mitten.

Input

The first line contains two integers n and m — the number of the children and the number of possible mitten colors ($1 \le n \le 5000$, $1 \le m \le 100$). The second line contains n integers $c_1, c_2, \dots c_n$, where c_i is the color of the mittens of the i-th child ($1 \le c_i \le m$).

Output

In the first line, print the maximum number of children who can end up with a distinct-colored pair of mittens. In the next n lines print the way the mittens can be distributed in this case. On the i-th of these lines print two space-separated integers: the color of the left and the color of the right mitten the i-th child will get. If there are multiple solutions, you can print any of them.

Sample test(s)

input
```
6 3
1 3 2 2 1 1
```
output
```
6
2 1
1 2
2 1
1 3
1 2
3 1
```

input
```
4 2
1 2 1 1
```
output
```
2
1 2
1 1
2 1
1 1
```

D. Broken Monitor

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

Innocentius has a problem — his computer monitor has broken. Now some of the pixels are "dead", that is, they are always black. As consequence, Innocentius can't play the usual computer games. He is recently playing the following game with his younger brother Polycarpus.

Innocentius is touch-typing a program that paints a white square one-pixel wide frame on the black screen. As the monitor is broken, some pixels that should be white remain black. Polycarpus should look at what the program displayed on the screen and guess the position and size of the frame Innocentius has painted. Polycarpus doesn't like the game but Innocentius persuaded brother to play as "the game is good for the imagination and attention".

Help Polycarpus, automatize his part in the gaming process. Write the code that finds such possible *square* frame that:

- the frame's width is 1 pixel,
- the frame doesn't go beyond the borders of the screen,
- all white pixels of the monitor are located on the frame,
- of all frames that satisfy the previous three conditions, the required frame must have the smallest size.

Formally, a square frame is represented by such pixels of the solid square, that are on the square's border, that is, are not fully surrounded by the other pixels of the square. For example, if the frame's size is $d = 3$, then it consists of 8 pixels, if its size is $d = 2$, then it contains 4 pixels and if $d = 1$, then the frame is reduced to a single pixel.

Input

The first line contains the resolution of the monitor as a pair of integers n, m ($1 \le n, m \le 2000$). The next n lines contain exactly m characters each — the state of the monitor pixels at the moment of the game. Character "`.`" (period, ASCII code 46) corresponds to the black pixel, and character "`w`" (lowercase English letter `w`) corresponds to the white pixel. It is guaranteed that at least one pixel of the monitor is white.

Output

Print the monitor screen. Represent the sought frame by characters "`+`" (the "plus" character). The pixels that has become white during the game mustn't be changed. Print them as "`w`". If there are multiple possible ways to position the frame of the minimum size, print any of them.

If the required frame doesn't exist, then print a single line containing number `-1`.

Sample test(s)

input
```
4 8
..w..w..
........
........
..w..w..
```
output
```
..w++w..
..+..+..
..+..+..
..w++w..
```

input
```
5 6
......
.w....
......
..w...
......
```
output
```
......
+w+...
+.+...
++w...
......
```

input
```
2 4
....
.w..
```
output
```
....
.w..
```

input
```
2 6
```

```
w..w.w
...w..
```

output

```
-1
```

Note

In the first sample the required size of the optimal frame equals 4. In the second sample the size of the optimal frame equals 3. In the third sample, the size of the optimal frame is 1. In the fourth sample, the required frame doesn't exist.

E. Summer Reading

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

At school Vasya got an impressive list of summer reading books. Unlike other modern schoolchildren, Vasya loves reading, so he read some book each day of the summer.

As Vasya was reading books, he was making notes in the Reader's Diary. Each day he wrote the orderal number of the book he was reading. The books in the list are numbered starting from 1 and Vasya was reading them in the order they go in the list. Vasya never reads a new book until he finishes reading the previous one. Unfortunately, Vasya wasn't accurate and some days he forgot to note the number of the book and the notes for those days remained empty.

As Vasya knows that the literature teacher will want to check the Reader's Diary, so he needs to restore the lost records. Help him do it and fill all the blanks. Vasya is sure that he spends at least two and at most five days for each book. Vasya finished reading all the books he had started. Assume that the reading list contained many books. So many, in fact, that it is impossible to read all of them in a summer. If there are multiple valid ways to restore the diary records, Vasya prefers the one that shows the maximum number of read books.

Input

The first line contains integer n — the number of summer days ($2 \le n \le 2 \cdot 10^5$). The second line contains n integers $a_1, a_2, \dots a_n$ — the records in the diary in the order they were written ($0 \le a_i \le 10^5$). If Vasya forgot to write the number of the book on the i-th day, then a_i equals 0.

Output

If it is impossible to correctly fill the blanks in the diary (the diary may contain mistakes initially), print "-1".

Otherwise, print in the first line the maximum number of books Vasya could have read in the summer if we stick to the diary. In the second line print n integers — the diary with correctly inserted records. If there are multiple optimal solutions, you can print any of them.

Sample test(s)

input

```
7
0 1 0 0 0 3 0
```

output

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

input

```
8
0 0 0 0 0 0 0 0
```

output

```
4
1 1 2 2 3 3 4 4
```

input

```
4
0 0 1 0
```

output

```
1
1 1 1 1
```

input

```
4
0 0 0 3
```

output

```
-1
```