Технокубок 2018 - Отборочный Раунд 1

A. k-rounding

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

For a given positive integer n denote its k-rounding as the minimum positive integer x, such that x ends with k or more zeros in base 10 and is divisible by n.

For example, 4-rounding of 375 is $375 \cdot 80 = 30000$. 30000 is the minimum integer such that it ends with 4 or more zeros and is divisible by 375.

Write a program that will perform the k-rounding of n.

Input
The only line contains two integers n and k ($1 \le n \le 10^9$, $0 \le k \le 8$).

Output
Print the k-rounding of n.

Examples

input
375 4
output
30000

input
10000 1
output
10000

input
38101 0
output
38101

input
123456789 8
output
12345678900000000

B. Which floor?

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

In a building where Polycarp lives there are *equal* number of flats on each floor. Unfortunately, Polycarp don't remember how many flats are on each floor, but he remembers that the flats are numbered from 1 from lower to upper floors. That is, the first several flats are on the first floor, the next several flats are on the second and so on. Polycarp don't remember the total number of flats in the building, so you can consider the building to be infinitely high (i.e. there are infinitely many floors). Note that the floors are numbered from 1.

Polycarp remembers on which floors several flats are located. It is guaranteed that this information is not self-contradictory. It means that there exists a building with equal number of flats on each floor so that the flats from Polycarp's memory have the floors Polycarp remembers.

Given this information, is it possible to restore the exact floor for flat n?

Input

The first line contains two integers n and m ($1 \le n \le 100$, $0 \le m \le 100$), where n is the number of the flat you need to restore floor for, and m is the number of flats in Polycarp's memory.

m lines follow, describing the Polycarp's memory: each of these lines contains a pair of integers k_i, f_i ($1 \le k_i \le 100$, $1 \le f_i \le 100$), which means that the flat k_i is on the f_i-th floor. All values k_i are distinct.

It is guaranteed that the given information is not self-contradictory.

Output

Print the number of the floor in which the n-th flat is located, if it is possible to determine it in a unique way. Print -1 if it is not possible to uniquely restore this floor.

Examples

input
10 3 6 2 2 1 7 3
output
4

input
8 4 3 1 6 2 5 2 2 1
output
-1

Note

In the first example the 6-th flat is on the 2-nd floor, while the 7-th flat is on the 3-rd, so, the 6-th flat is the last on its floor and there are 3 flats on each floor. Thus, the 10-th flat is on the 4-th floor.

In the second example there can be 3 or 4 flats on each floor, so we can't restore the floor for the 8-th flat.

C. Did you mean...

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

Beroffice text editor has a wide range of features that help working with text. One of the features is an automatic search for typos and suggestions of how to fix them.

Beroffice works only with small English letters (i.e. with 26 letters from `a` to `z`). Beroffice thinks that a word is typed with a typo if there are three or more consonants in a row in the word. The only exception is that if the block of consonants has all letters the same, then this block (even if its length is greater than three) is not considered a typo. Formally, a word is typed with a typo if there is a block of not less that three consonants in a row, and there are at least two different letters in this block.

For example:

- the following words have typos: "`hellno`", "`hackcerrs`" and "`backtothefuttttture`";
- the following words don't have typos: "`hellllllooooo`", "`tobeornottobe`" and "`oooooo`".

When Beroffice editor finds a word with a typo, it inserts as little as possible number of spaces in this word (dividing it into several words) in such a way that each of the resulting words is typed without any typos.

Implement this feature of Beroffice editor. Consider the following letters as the only vowels: '`a`', '`e`', '`i`', '`o`' and '`u`'. All the other letters are consonants in this problem.

Input

The only line contains a non-empty word consisting of small English letters. The length of the word is between 1 and 3000 letters.

Output

Print the given word without any changes if there are no typos.

If there is at least one typo in the word, insert the minimum number of spaces into the word so that each of the resulting words doesn't have any typos. If there are multiple solutions, print any of them.

Examples

input
hellno
output
hell no

input
abacaba
output
abacaba

input
asdfasdf
output
asd fasd f

D. Polycarp's phone book

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n phone numbers in Polycarp's contacts on his phone. Each number is a 9-digit integer, starting with a digit different from 0. All the numbers are distinct.

There is the latest version of Berdroid OS installed on Polycarp's phone. If some number is entered, is shows up all the numbers in the contacts for which there is a substring equal to the entered sequence of digits. For example, is there are three phone numbers in Polycarp's contacts: 123456789, 100000000 and 100123456, then:

- if he enters 00 two numbers will show up: 100000000 and 100123456,
- if he enters 123 two numbers will show up 123456789 and 100123456,
- if he enters 01 there will be only one number 100123456.

For each of the phone numbers in Polycarp's contacts, find the minimum in length sequence of digits such that if Polycarp enters this sequence, Berdroid shows this only phone number.

Input

The first line contains single integer n ($1 \le n \le 70000$) — the total number of phone contacts in Polycarp's contacts.

The phone numbers follow, one in each line. Each number is a positive 9-digit integer starting with a digit from 1 to 9. All the numbers are distinct.

Output

Print exactly n lines: the i-th of them should contain the shortest non-empty sequence of digits, such that if Polycarp enters it, the Berdroid OS shows up only the i-th number from the contacts. If there are several such sequences, print any of them.

Examples

input
3 123456789 100000000 100123456
output
9 000 01

input
4 123456789 193456789 134567819 934567891
output
2 193 81 91

E. Tests Renumeration

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

The All-Berland National Olympiad in Informatics has just ended! Now Vladimir wants to upload the contest from the Olympiad as a gym to a popular Codehorses website.

Unfortunately, the archive with Olympiad's data is a mess. For example, the files with tests are named arbitrary without any logic.

Vladimir wants to rename the files with tests so that their names are distinct integers starting from 1 without any gaps, namely, "`1`", "`2`", ..., "n', where n is the total number of tests.

Some of the files contain tests from statements (examples), while others contain regular tests. It is possible that there are no examples, and it is possible that all tests are examples. Vladimir wants to rename the files so that the examples are the first several tests, all all the next files contain regular tests only.

The only operation Vladimir can perform is the "`move`" command. Vladimir wants to write a script file, each of the lines in which is "`move file_1 file_2`", that means that the file "`file_1`" is to be renamed to "`file_2`". If there is a file "`file_2`" at the moment of this line being run, then this file is to be rewritten. After the line "`move file_1 file_2`" the file "`file_1`" doesn't exist, but there is a file "`file_2`" with content equal to the content of "`file_1`" before the "`move`" command.

Help Vladimir to write the script file with the minimum possible number of lines so that after this script is run:

- all examples are the first several tests having filenames "`1`", "`2`", ..., "e", where e is the total number of examples;
- all other files contain regular tests with filenames "$e + 1$", "$e + 2$", ..., "n", where n is the total number of all tests.

Input

The first line contains single integer n ($1 \le n \le 10^5$) — the number of files with tests.

n lines follow, each describing a file with test. Each line has a form of "`name_i type_i`", where "`name_i`" is the filename, and "`type_i`" equals "`1`", if the i-th file contains an example test, and "`0`" if it contains a regular test. Filenames of each file are strings of digits and small English letters with length from 1 to 6 characters. The filenames are guaranteed to be distinct.

Output

In the first line print the minimum number of lines in Vladimir's script file.

After that print the script file, each line should be "`move file_1 file_2`", where "`file_1`" is an existing at the moment of this line being run filename, and "`file_2`" — is a string of digits and small English letters with length from 1 to 6.

Examples

input
5 01 0 2 1 2extra 0 3 1 99 0
output
4 move 3 1 move 01 5 move 2extra 4 move 99 3

input
2 1 0 2 1
output
3 move 1 3 move 2 1 move 3 2

input
5 1 0 11 1 111 0 1111 1 11111 0
output

```
5
move 1 5
move 11 1
move 1111 2
move 111 4
move 11111 3
```

F. Wizard's Tour

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

All Berland residents are waiting for an unprecedented tour of wizard in his Blue Helicopter over the cities of Berland!

It is well-known that there are n cities in Berland, some pairs of which are connected by bidirectional roads. Each pair of cities is connected by no more than one road. It is not guaranteed that the road network is *connected*, i.e. it is possible that you can't reach some city from some other.

The tour will contain several episodes. In each of the episodes:

- the wizard will disembark at some city x from the Helicopter;
- he will give a performance and show a movie for free at the city x;
- he will drive to some neighboring city y using a road;
- he will give a performance and show a movie for free at the city y;
- he will drive to some neighboring to y city z;
- he will give a performance and show a movie for free at the city z;
- he will embark the Helicopter and fly away from the city z.

It is known that the wizard doesn't like to use roads, so he agrees to use each road at most once (regardless of direction). In other words, for road between a and b he only can drive once from a to b, or drive once from b to a, or do not use this road at all.

The wizards wants to plan as many episodes as possible without violation the above rules. Help the wizard!

Please note that the wizard can visit the same city multiple times, the restriction is on roads only.

Input

The first line contains two integers n, m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — the number of cities and the number of roads in Berland, respectively.

The roads description follow, one in each line. Each description is a pair of two integers a_i, b_i ($1 \le a_i$, $b_i \le n$, $a_i \ne b_i$), where a_i and b_i are the ids of the cities connected by the i-th road. It is guaranteed that there are no two roads connecting the same pair of cities. Every road is bidirectional. The cities are numbered from 1 to n.

It is possible that the road network in Berland is not connected.

Output

In the first line print w — the maximum possible number of episodes. The next w lines should contain the episodes in format x, y, z — the three integers denoting the ids of the cities in the order of the wizard's visits.

Examples

input
4 5 1 2 3 2 2 4 3 4 4 1
output
2 1 4 2 4 3 2

input
5 8 5 3 1 2 4 5 5 1 2 5 4 3 1 4 3 2
output
4 1 4 5 2 3 4 1 5 3 5 2 1