A. Free Cash

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

Valera runs a 24/7 fast food cafe. He magically learned that next day n people will visit his cafe. For each person we know the arrival time: the i-th person comes exactly at h_i hours m_i minutes. The cafe spends less than a minute to serve each client, but if a client comes in and sees that there is no free cash, than he doesn't want to wait and leaves the cafe immediately.

Valera is very greedy, so he wants to serve all n customers next day (and get more profit). However, for that he needs to ensure that at each moment of time the number of working cashes is no less than the number of clients in the cafe.

Help Valera count the minimum number of cashes to work at his cafe next day, so that they can serve all visitors.

Input

The first line contains a single integer n $(1 \le n \le 10^5)$, that is the number of cafe visitors.

Each of the following n lines has two space-separated integers h_i and m_i $(0 \le h_i \le 23;\ 0 \le m_i \le 59)$, representing the time when the i-th person comes into the cafe.

Note that the time is given in the **chronological** order. All time is given within one 24-hour period.

Output

Print a single integer — the minimum number of cashes, needed to serve all clients next day.

Sample test(s)

input

```
4
8 0
8 10
8 10
8 45
```

output

```
2
```

input

```
3
0 12
10 11
22 22
```

output

```
1
```

Note

In the first sample it is not enough one cash to serve all clients, because two visitors will come into cafe in 8:10. Therefore, if there will be one cash in cafe, then one customer will be served by it, and another one will not wait and will go away.

In the second sample all visitors will come in different times, so it will be enough one cash.

B. Young Table

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

You've got table a, consisting of n rows, numbered from 1 to n. The i-th line of table a contains c_i cells, at that for all i $(1 < i \le n)$ holds $c_i \le c_{i-1}$.

Let's denote s as the total number of cells of table a, that is, $s = \sum_{i=1}^{n} c_i$. We know that each cell of the table contains a single integer from 1 to s, at that all written integers are distinct.

Let's assume that the cells of the i-th row of table a are numbered from 1 to c_i, then let's denote the number written in the j-th cell of the i-th row as $a_{i,j}$. Your task is to perform several swap operations to rearrange the numbers in the table so as to fulfill the following conditions:

1. for all i, j $(1 < i \le n;\ 1 \le j \le c_i)$ holds $a_{i,j} > a_{i-1,j}$;
2. for all i, j $(1 \le i \le n;\ 1 < j \le c_i)$ holds $a_{i,j} > a_{i,j-1}$.

In one swap operation you are allowed to choose two different cells of the table and swap the recorded there numbers, that is the number that was recorded in the first of the selected cells before the swap, is written in the second cell after it. Similarly, the number that was recorded in the second of the selected cells, is written in the first cell after the swap.

Rearrange the numbers in the required manner. Note that you are allowed to perform any number of operations, but not more than s. You do not have to minimize the number of operations.

Input

The first line contains a single integer n $(1 \le n \le 50)$ that shows the number of rows in the table. The second line contains n space-separated integers c_i $(1 \le c_i \le 50;\ c_i \le c_{i-1})$ — the numbers of cells on the corresponding rows.

Next n lines contain table a. The i-th of them contains c_i space-separated integers: the j-th integer in this line represents $a_{i,j}$.

It is guaranteed that all the given numbers $a_{i,j}$ are positive and do not exceed s. It is guaranteed that all $a_{i,j}$ are distinct.

Output

In the first line print a single integer m $(0 \le m \le s)$, representing the number of performed swaps.

In the next m lines print the description of these swap operations. In the i-th line print four space-separated integers x_i, y_i, p_i, q_i $(1 \le x_i, p_i \le n;\ 1 \le y_i \le c_{x_i};\ 1 \le q_i \le c_{p_i})$. The printed numbers denote swapping the contents of cells a_{x_i, y_i} and a_{p_i, q_i}. Note that a swap operation can change the contents of **distinct** table cells. Print the swaps in the order, in which they should be executed.

Sample test(s)

input
```
3
3 2 1
4 3 5
6 1
2
```
output
```
2
1 1 2 2
2 1 3 1
```

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

C. Primes on Interval

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

You've decided to carry out a survey in the theory of prime numbers. Let us remind you that a prime number is a positive integer that has exactly two distinct positive integer divisors.

Consider positive integers a, $a + 1$, ..., b $(a \le b)$. You want to find the minimum integer l $(1 \le l \le b - a + 1)$ such that for any integer x $(a \le x \le b - l + 1)$ among l integers x, $x + 1$, ..., $x + l - 1$ there are at least k prime numbers.

Find and print the required minimum l. If no value l meets the described limitations, print -1.

Input

A single line contains three space-separated integers a, b, k $(1 \le a, b, k \le 10^6;\ a \le b)$.

Output

In a single line print a single integer — the required minimum l. If there's no solution, print -1.

Sample test(s)

input

```
2 4 2
```

output

```
3
```

input

```
6 13 1
```

output

```
4
```

input

```
1 4 3
```

output

```
-1
```

D. T-decomposition

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

You've got a undirected tree s, consisting of n nodes. Your task is to build an optimal T-decomposition for it. Let's define a T-decomposition as follows.

Let's denote the set of all nodes s as v. Let's consider an undirected tree t, whose nodes are some non-empty subsets of v, we'll call them x_i $(x_i \subseteq v)$. The tree t is a T-decomposition of s, if the following conditions holds:

1. the union of all x_i equals v;
2. for any edge (a, b) of tree s exists the tree node t, containing both a and b;
3. if the nodes of the tree t x_i and x_j contain the node a of the tree s, then all nodes of the tree t, lying on the path from x_i to x_j also contain node a. So this condition is equivalent to the following: all nodes of the tree t, that contain node a of the tree s, form a connected subtree of tree t.

There are obviously many distinct trees t, that are T-decompositions of the tree s. For example, a T-decomposition is a tree that consists of a single node, equal to set v.

Let's define the cardinality of node x_i as the number of nodes in tree s, containing in the node. Let's choose the node with the maximum cardinality in t. Let's assume that its cardinality equals w. Then the weight of T-decomposition t is value w. The optimal T-decomposition is the one with the minimum weight.

Your task is to find the optimal T-decomposition of the given tree s that has the minimum number of nodes.

Input

The first line contains a single integer n $(2 \le n \le 10^5)$, that denotes the number of nodes in tree s.

Each of the following $n - 1$ lines contains two space-separated integers a_i, b_i $(1 \le a_i, b_i \le n; a_i \ne b_i)$, denoting that the nodes of tree s with indices a_i and b_i are connected by an edge.

Consider the nodes of tree s indexed from 1 to n. It is guaranteed that s is a tree.

Output

In the first line print a single integer m that denotes the number of nodes in the required T-decomposition.

Then print m lines, containing descriptions of the T-decomposition nodes. In the i-th $(1 \le i \le m)$ of them print the description of node x_i of the T-decomposition. The description of each node x_i should start from an integer k_i, that represents the number of nodes of the initial tree s, that are contained in the node x_i. Then you should print k_i distinct space-separated integers — the numbers of nodes from s, contained in x_i, in arbitrary order.

Then print $m - 1$ lines, each consisting two integers p_i, q_i $(1 \le p_i, q_i \le m; p_i \ne q_i)$. The pair of integers p_i, q_i means there is an edge between nodes x_{p_i} and x_{q_i} of T-decomposition.

The printed T-decomposition should be the optimal T-decomposition for the given tree s and have the minimum possible number of nodes among all optimal T-decompositions. If there are multiple optimal T-decompositions with the minimum number of nodes, print any of them.

Sample test(s)

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

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

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

```
output
```

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

E. Build String

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

You desperately need to build some string t. For that you've got n more strings $s_1, s_2, ..., s_n$. To build string t, you are allowed to perform exactly $|t|$ ($|t|$ is the length of string t) operations on these strings. Each operation looks like that:

1. choose any non-empty string from strings $s_1, s_2, ..., s_n$;
2. choose an arbitrary character from the chosen string and write it on a piece of paper;
3. remove the chosen character from the chosen string.

Note that after you perform the described operation, the total number of characters in strings $s_1, s_2, ..., s_n$ decreases by 1. We are assumed to build string t, if the characters, written on the piece of paper, in the order of performed operations form string t.

There are other limitations, though. For each string s_i you know number a_i — the maximum number of characters you are allowed to delete from string s_i. You also know that each operation that results in deleting a character from string s_i, costs i rubles. That is, an operation on string s_1 is the cheapest (it costs 1 ruble), and the operation on string s_n is the most expensive one (it costs n rubles).

Your task is to count the minimum amount of money (in rubles) you will need to build string t by the given rules. Consider the cost of building string t to be the sum of prices of the operations you use.

Input

The first line of the input contains string t — the string that you need to build.

The second line contains a single integer n $(1 \le n \le 100)$ — the number of strings to which you are allowed to apply the described operation. Each of the next n lines contains a string and an integer. The i-th line contains space-separated string s_i and integer a_i $(0 \le a_i \le 100)$. Number a_i represents the maximum number of characters that can be deleted from string s_i.

All strings in the input only consist of lowercase English letters. All strings are non-empty. The lengths of all strings do not exceed 100 characters.

Output

Print a single number — the minimum money (in rubles) you need in order to build string t. If there is no solution, print -1.

Sample test(s)

input

```
bbaze
3
bzb 2
aeb 3
ba 10
```

output

```
8
```

input

```
abacaba
4
aba 2
bcc 1
caa 2
bbb 5
```

output

```
18
```

input

```
xyz
4
axx 8
za 1
efg 4
t 1
```

output

```
-1
```

Note

Notes to the samples:

In the first sample from the first string you should take characters "b" and "z" with price 1 ruble, from the second string characters "a", "e" и "b" with price 2 rubles. The price of the string t in this case is $2 \cdot 1 + 3 \cdot 2 = 8$.

In the second sample from the first string you should take two characters "a" with price 1 ruble, from the second string character "c" with price 2

rubles, from the third string two characters "a" with price 3 rubles, from the fourth string two characters "b" with price 4 rubles. The price of the string t in this case is $2 \cdot 1 + 1 \cdot 2 + 2 \cdot 3 + 2 \cdot 4 = 18$.

In the third sample the solution doesn't exist because there is no character "y" in given strings.