A. Design Tutorial: Learn from Math

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

One way to create a task is to learn from math. You can generate some random math statement or modify some theorems to get something new and build a new task from that.

For example, there is a statement called the "Goldbach's conjecture". It says: "each even number no less than four can be expressed as the sum of two primes". Let's modify it. How about a statement like that: "each integer no less than 12 can be expressed as the sum of two composite numbers." Not like the Goldbach's conjecture, I can prove this theorem.

You are given an integer n no less than 12, express it as a sum of two composite numbers.

Input

The only line contains an integer n $(12 \le n \le 10^6)$.

Output

Output two composite integers x and y $(1 < x, y < n)$ such that $x + y = n$. If there are multiple solutions, you can output any of them.

Sample test(s)

input
```
12
```
output
```
4 8
```

input
```
15
```
output
```
6 9
```

input
```
23
```
output
```
8 15
```

input
```
1000000
```
output
```
500000 500000
```

Note

In the first example, 12 = 4 + 8 and both 4, 8 are composite numbers. You can output "6 6" or "8 4" as well.

In the second example, 15 = 6 + 9. Note that you can't output "1 14" because 1 is not a composite number.

B. Design Tutorial: Learn from Life

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

One way to create a task is to learn from life. You can choose some experience in real life, formalize it and then you will get a new task.

Let's think about a scene in real life: there are lots of people waiting in front of the elevator, each person wants to go to a certain floor. We can formalize it in the following way. We have n people standing on the first floor, the i-th person wants to go to the f_i-th floor. Unfortunately, there is only one elevator and its capacity equal to k (that is at most k people can use it simultaneously). Initially the elevator is located on the first floor. The elevator needs $|a - b|$ seconds to move from the a-th floor to the b-th floor (we don't count the time the people need to get on and off the elevator).

What is the minimal number of seconds that is needed to transport all the people to the corresponding floors and then return the elevator to the first floor?

Input

The first line contains two integers n and k $(1 \le n, k \le 2000)$ — the number of people and the maximal capacity of the elevator.

The next line contains n integers: $f_1, f_2, ..., f_n$ $(2 \le f_i \le 2000)$, where f_i denotes the target floor of the i-th person.

Output

Output a single integer — the minimal time needed to achieve the goal.

Sample test(s)

input

```
3 2
2 3 4
```

output

```
8
```

input

```
4 2
50 100 50 100
```

output

```
296
```

input

```
10 3
2 2 2 2 2 2 2 2 2 2
```

output

```
8
```

Note

In first sample, an optimal solution is:

1. The elevator takes up person #1 and person #2.
2. It goes to the 2nd floor.
3. Both people go out of the elevator.
4. The elevator goes back to the 1st floor.
5. Then the elevator takes up person #3.
6. And it goes to the 2nd floor.
7. It picks up person #2.
8. Then it goes to the 3rd floor.
9. Person #2 goes out.
10. Then it goes to the 4th floor, where person #3 goes out.
11. The elevator goes back to the 1st floor.

C. Design Tutorial: Make It Nondeterministic

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

A way to make a new task is to make it nondeterministic or probabilistic. For example, the hard task of Topcoder SRM 595, Constellation, is the probabilistic version of a convex hull.

Let's try to make a new task. Firstly we will use the following task. There are n people, sort them by their name. It is just an ordinary sorting problem, but we can make it more interesting by adding nondeterministic element. There are n people, each person will use either his/her first name or last name as a handle. Can the lexicographical order of the handles be exactly equal to the given permutation p?

More formally, if we denote the handle of the i-th person as h_i, then the following condition must hold: $\forall\ i, j\ (i < j):\ h_{p_i} < h_{p_j}$.

Input

The first line contains an integer n $(1 \le n \le 10^5)$ — the number of people.

The next n lines each contains two strings. The i-th line contains strings f_i and s_i $(1 \le |f_i|, |s_i| \le 50)$ — the first name and last name of the i-th person. Each string consists only of lowercase English letters. All of the given $2n$ strings will be distinct.

The next line contains n distinct integers: $p_1, p_2, ..., p_n$ $(1 \le p_i \le n)$.

Output

If it is possible, output "`YES`", otherwise output "`NO`".

Sample test(s)

input

```
3
gennady korotkevich
petr mitrichev
gaoyuan chen
1 2 3
```

output

```
NO
```

input

```
3
gennady korotkevich
petr mitrichev
gaoyuan chen
3 1 2
```

output

```
YES
```

input

```
2
galileo galilei
nicolaus copernicus
2 1
```

output

```
YES
```

input

```
10
rean schwarzer
fei claussell
alisa reinford
eliot craig
laura arseid
jusis albarea
machias regnitz
sara valestin
emma millstein
gaius worzel
1 2 3 4 5 6 7 8 9 10
```

output

```
NO
```

input

```
10
rean schwarzer
```

```
fei claussell
alisa reinford
eliot craig
laura arseid
jusis albarea
machias regnitz
sara valestin
emma millstein
gaius worzel
2 4 9 6 5 7 1 3 8 10
```

output

```
YES
```

Note

In example 1 and 2, we have 3 people: tourist, Petr and me (cgy4ever). You can see that whatever handle is chosen, I must be the first, then tourist and Petr must be the last.

In example 3, if Copernicus uses "copernicus" as his handle, everything will be alright.

D. Design Tutorial: Inverse the Problem

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

There is an easy way to obtain a new task from an old one called "Inverse the problem": we give an output of the original task, and ask to generate an input, such that solution to the original problem will produce the output we provided. The hard task of Topcoder Open 2014 Round 2C, InverseRMQ, is a good example.

Now let's create a task this way. We will use the task: you are given a tree, please calculate the distance between any pair of its nodes. Yes, it is very easy, but the inverse version is a bit harder: you are given an $n \times n$ distance matrix. Determine if it is the distance matrix of a weighted tree (all weights must be positive integers).

Input

The first line contains an integer n $(1 \le n \le 2000)$ — the number of nodes in that graph.

Then next n lines each contains n integers $d_{i,j}$ $(0 \le d_{i,j} \le 10^9)$ — the distance between node i and node j.

Output

If there exists such a tree, output "`YES`", otherwise output "`NO`".

Sample test(s)

input

```
3
0 2 7
2 0 9
7 9 0
```

output

```
YES
```

input

```
3
1 2 7
2 0 9
7 9 0
```

output

```
NO
```

input

```
3
0 2 2
7 0 9
7 9 0
```

output

```
NO
```

input

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

output

```
NO
```

input

```
2
0 0
0 0
```

output

```
NO
```

Note

In the first example, the required tree exists. It has one edge between nodes 1 and 2 with weight 2, another edge between nodes 1 and 3 with weight 7.

In the second example, it is impossible because $d_{1,1}$ should be 0, but it is 1.

In the third example, it is impossible because $d_{1,2}$ should equal $d_{2,1}$.

E. Design Tutorial: Learn from a Game

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

One way to create task is to learn from game. You should pick a game and focus on part of the mechanic of that game, then it might be a good task.

Let's have a try. Puzzle and Dragon was a popular game in Japan, we focus on the puzzle part of that game, it is a tile-matching puzzle.

(Picture from Wikipedia page: http://en.wikipedia.org/wiki/Puzzle_&_Dragons)

There is an $n \times m$ board which consists of orbs. During the game you can do the following move. In the beginning of move you touch a cell of the board, then you can move your finger to one of the adjacent cells (a cell not on the boundary has 8 adjacent cells), then you can move your finger from the current cell to one of the adjacent cells one more time, and so on. Each time you move your finger from a cell to another cell, the orbs in these cells swap with each other. In other words whatever move you make, the orb in the cell you are touching never changes.

The goal is to achieve such kind of pattern that the orbs will be cancelled and your monster will attack the enemy, but we don't care about these details. Instead, we will give you the initial board as an input and the target board as an output. Your goal is to determine whether there is a way to reach the target in a single move.

Input

The first line contains two integers: n and m $(1 \le n, m \le 30)$.

The next n lines each contains m integers — the description of the initial board. The j-th integer in the i-th line is $s_{i,j}$ $(1 \le s_{i,j} \le 900)$, where $s_{i,j}$ denotes the type of the orb located in the i-th row and j-th column of the board.

The next n lines contain the target board in the same format. Note, that the initial board and the target board will be different.

Output

If there is no solution, then output: `-1`.

If there is a solution, then in the first line output an integer k $(1 \le k \le 10^6)$ — the number of finger moves.

In the next line print two integers x_0 and y_0 $(1 \le x_0 \le n;\ 1 \le y_0 \le m)$ — the position of the cell you touch at the beginning. In each of the next k lines print two integers x_i and y_i $(1 \le x_i \le n;\ 1 \le y_i \le m)$ — the position you move to. Note that this position must be adjacent to the previous position, that is $max(|x_i - x_{i-1}|, |y_i - y_{i-1}|) = 1$.

If there are multiple solutions, you can print any of them. We can prove that under these constraints if there exists a solution then there is a solution with no more than 10^6 operations.

Sample test(s)

input

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

output

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

input

```
2 2
```

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

output

```
-1
```

input

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

output

```
-1
```

input

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

output

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

F. Design Tutorial: Change the Goal

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

There are some tasks which have the following structure: you are given a model, and you can do some operations, you should use these operations to achive the goal. One way to create a new task is to use the same model and same operations, but change the goal.

Let's have a try. I have created the following task for Topcoder SRM 557 Div1-Hard: you are given n integers $x_1, x_2, ..., x_n$. You are allowed to perform the assignments (as many as you want) of the following form x_i ^= x_j (in the original task i and j must be different, but in this task we allow i to equal j). The goal is to maximize the sum of all x_i.

Now we just change the goal. You are also given n integers $y_1, y_2, ..., y_n$. You should make $x_1, x_2, ..., x_n$ exactly equal to $y_1, y_2, ..., y_n$. In other words, for each i number x_i should be equal to y_i.

Input

The first line contains an integer n $(1 \le n \le 10000)$. The second line contains n integers: x_1 to x_n $(0 \le x_i \le 10^9)$. The third line contains n integers: y_1 to y_n $(0 \le y_i \le 10^9)$.

Output

If there is no solution, output `-1`.

If there is a solution, then in the first line output an integer m $(0 \le m \le 1000000)$ – the number of assignments you need to perform. Then print m lines, each line should contain two integers i and j $(1 \le i, j \le n)$, which denote assignment x_i ^= x_j.

If there are multiple solutions you can print any of them. We can prove that under these constraints if there exists a solution then there always exists a solution with no more than 10^6 operations.

Sample test(s)

input
```
2
3 5
6 0
```
output
```
2
1 2
2 2
```

input
```
5
0 0 0 0 0
1 2 3 4 5
```
output
```
-1
```

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

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

Note

Assignment a ^= b denotes assignment $a = a$ ^ b, where operation "^" is bitwise XOR of two integers.

G. Design Tutorial: Increase the Constraints

time limit per test: 7 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a simple way to create hard tasks: take one simple problem as the query, and try to find an algorithm that can solve it faster than bruteforce. This kind of tasks usually appears in OI contest, and usually involves data structures.

Let's try to create a task, for example, we take the "Hamming distance problem": for two binary strings s and t with the same length, the Hamming distance between them is the number of positions at which the corresponding symbols are different. For example, the Hamming distance between "**0**01**1**1" and "**1**01**0**1" is 2 (the different symbols are marked with bold).

We use the Hamming distance problem as a query in the following way: you are given two strings a and b and several queries. Each query will be: what is the Hamming distance between two strings $a_{p_1}a_{p_1+1}...a_{p_1+len-1}$ and $b_{p_2}b_{p_2+1}...b_{p_2+len-1}$?

Note, that in this problem the strings are **zero-based**, that is $s = s_0s_1... s_{|s|-1}$.

Input

The first line contains a string a $(1 \le |a| \le 200000)$. The second line contains a string b $(1 \le |b| \le 200000)$. Each character of both strings is either "0" or "1".

The third line contains an integer q $(1 \le q \le 400000)$ — the number of queries. Each of the following q lines contains three integers: p_1, p_2 and len $(0 \le p_1 \le |a| - len;\ 0 \le p_2 \le |b| - len)$, these numbers denote the parameters of the current query.

Output

Output q integers — the answers for the queries.

Sample test(s)

input

```
101010
11110000
3
0 0 3
2 3 4
5 7 1
```

output

```
1
1
0
```

input

```
10001010101011001010100101010011010
101010100101001010100100101010
5
0 0 12
3 9 7
6 4 15
12 15 10
13 3 20
```

output

```
5
4
3
5
13
```