A. Sushi for Two

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

Arkady invited Anna for a dinner to a sushi restaurant. The restaurant is a bit unusual: it offers n pieces of sushi aligned in a row, and a customer has to choose a continuous subsegment of these sushi to buy.

The pieces of sushi are of two types: either with tuna or with eel. Let's denote the type of the i-th from the left sushi as t_i, where $t_i = 1$ means it is with tuna, and $t_i = 2$ means it is with eel.

Arkady does not like tuna, Anna does not like eel. Arkady wants to choose such a continuous subsegment of sushi that it has equal number of sushi of each type and each half of the subsegment has only sushi of one type. For example, subsegment $[2, 2, 2, 1, 1, 1]$ is valid, but subsegment $[1, 2, 1, 2, 1, 2]$ is not, because both halves contain both types of sushi.

Find the length of the longest continuous subsegment of sushi Arkady can buy.

Input

The first line contains a single integer n ($2 \le n \le 100\,000$) — the number of pieces of sushi.

The second line contains n integers t_1, t_2, ..., t_n ($t_i = 1$, denoting a sushi with tuna or $t_i = 2$, denoting a sushi with eel), representing the types of sushi from left to right.

It is guaranteed that there is at least one piece of sushi of each type. Note that it means that there is at least one valid continuous segment.

Output

Print a single integer — the maximum length of a valid continuous segment.

Examples

input

```
7
2 2 2 1 1 2 2
```

output

```
4
```

input

```
6
1 2 1 2 1 2
```

output

```
2
```

input

```
9
2 2 1 1 1 2 2 2 2
```

output

```
6
```

Note

In the first example Arkady can choose the subsegment $[2, 2, 1, 1]$ or the subsegment $[1, 1, 2, 2]$ with length 4.

In the second example there is no way but to choose one of the subsegments $[2, 1]$ or $[1, 2]$ with length 2.

In the third example Arkady's best choice is the subsegment $[1, 1, 1, 2, 2, 2]$.

B. Circus

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

Polycarp is a head of a circus troupe. There are n — an even number — artists in the troupe. It is known whether the i-th artist can perform as a clown (if yes, then $c_i = 1$, otherwise $c_i = 0$), and whether they can perform as an acrobat (if yes, then $a_i = 1$, otherwise $a_i = 0$).

Split the artists into two performances in such a way that:

- each artist plays in exactly one performance,
- the number of artists in the two performances is equal (i.e. equal to $\frac{n}{2}$),
- the number of artists that can perform as clowns in the first performance is the same as the number of artists that can perform as acrobats in the second performance.

Input

The first line contains a single integer n ($2 \le n \le 5\,000$, n is even) — the number of artists in the troupe.

The second line contains n digits $c_1 c_2 \ldots c_n$, the i-th of which is equal to 1 if the i-th artist can perform as a clown, and 0 otherwise.

The third line contains n digits $a_1 a_2 \ldots a_n$, the i-th of which is equal to 1, if the i-th artist can perform as an acrobat, and 0 otherwise.

Output

Print $\frac{n}{2}$ distinct integers — the indices of the artists that should play in the first performance.

If there are multiple answers, print any.

If there is no solution, print a single integer -1.

Examples

input

```
4
0011
0101
```

output

```
1 4
```

input

```
6
000000
111111
```

output

```
-1
```

input

```
4
0011
1100
```

output

```
4 3
```

input

```
8
00100101
01111100
```

output

```
1 2 3 6
```

Note

In the first example, one of the possible divisions into two performances is as follows: in the first performance artists 1 and 4 should take part. Then the number of artists in the first performance who can perform as clowns is equal to 1. And the number of artists in the second performance who can perform as acrobats is 1 as well.

In the second example, the division is not possible.

In the third example, one of the possible divisions is as follows: in the first performance artists 3 and 4 should take part. Then in the first performance there are 2 artists who can perform as clowns. And the number of artists in the second performance who can perform as acrobats is 2 as well.

C. Skyscrapers

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Dora loves adventures quite a lot. During some journey she encountered an amazing city, which is formed by n streets along the

Eastern direction and m streets across the Southern direction. Naturally, this city has nm intersections. At any intersection of i-th Eastern street and j-th Southern street there is a monumental skyscraper. Dora instantly became curious and decided to explore the heights of the city buildings.

When Dora passes through the intersection of the i-th Eastern and j-th Southern street she examines those two streets. After Dora learns the heights of all the skyscrapers on those two streets she wonders: how one should reassign heights to the skyscrapers on those two streets, so that the maximum height would be as small as possible and the result of comparing the heights of any two skyscrapers on one street wouldn't change.

Formally, on every of nm intersections Dora solves an independent problem. She sees $n + m - 1$ skyscrapers and for each of them she knows its real height. Moreover, any two heights can be compared to get a result "greater", "smaller" or "equal". Now Dora wants to select some integer x and assign every skyscraper a height from 1 to x. When assigning heights, Dora wants to preserve the relative order of the skyscrapers in both streets. That is, the result of any comparison of heights of two skyscrapers in the current Eastern street shouldn't change and the result of any comparison of heights of two skyscrapers in current Southern street shouldn't change as well. Note that skyscrapers located on the Southern street are not compared with skyscrapers located on the Eastern street only. However, the skyscraper located at the streets intersection can be compared with both Southern and Eastern skyscrapers. For every intersection Dora wants to **independently** calculate the minimum possible x.

For example, if the intersection and the two streets corresponding to it look as follows:

$$\boxed{\begin{array}{ccccc} & & 10 & & \\ & & 20 & & \\ 11 & 13 & 14 & 15 & 16 \\ & & 10 & & \end{array}}$$

Then it is optimal to replace the heights of the skyscrapers as follows (note that all comparisons "less", "equal", "greater" inside the Eastern street and inside the Southern street are preserved)

$$\boxed{\begin{array}{ccccc} & & 1 & & \\ & & 4 & & \\ 1 & 2 & 3 & 4 & 5 \\ & & 1 & & \end{array}}$$

The largest used number is 5, hence the answer for this intersection would be 5.

Help Dora to compute the answers for each intersection.

Input

The first line contains two integers n and m ($1 \le n, m \le 1000$) — the number of streets going in the Eastern direction and the number of the streets going in Southern direction.

Each of the following n lines contains m integers $a_{i,1}$, $a_{i,2}$, ..., $a_{i,m}$ ($1 \le a_{i,j} \le 10^9$). The integer $a_{i,j}$, located on j-th position in the i-th line denotes the height of the skyscraper at the intersection of the i-th Eastern street and j-th Southern direction.

Output

Print n lines containing m integers each. The integer $x_{i,j}$, located on j-th position inside the i-th line is an answer for the problem at the intersection of i-th Eastern street and j-th Southern street.

Examples

input

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

output

```
2 2 2
2 2 2
```

input

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

output

```
2 3
3 2
```

Note

In the first example, it's not possible to decrease the maximum used height for the problem at any intersection, hence we don't have to change any heights.

In the second example, the answers are as follows:

- For the intersection of the first line and the first column

$$\boxed{\begin{array}{cc} 1 & 2 \\ 3 & \end{array}} \to \boxed{\begin{array}{cc} 1 & 2 \\ 2 & \end{array}}$$

- For the intersection of the first line and the second column

$$\boxed{\begin{array}{cc} 1 & 2 \\ & 4 \end{array}} \to \boxed{\begin{array}{cc} 1 & 2 \\ & 3 \end{array}}$$

- For the intersection of the second line and the first column

$$\boxed{\begin{array}{cc} 1 & \\ 3 & 4 \end{array}} \to \boxed{\begin{array}{cc} 1 & \\ 2 & 3 \end{array}}$$

- For the intersection of the second line and the second column

$$\boxed{\begin{array}{cc} & 2 \\ 3 & 4 \end{array}} \to \boxed{\begin{array}{cc} & 1 \\ 1 & 2 \end{array}}$$

D. Camp Schedule

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

The new camp by widely-known over the country Spring Programming Camp is going to start soon. Hence, all the team of friendly curators and teachers started composing the camp's schedule. After some continuous discussion, they came up with a schedule s, which can be represented as a binary string, in which the i-th symbol is '1' if students will write the contest in the i-th day and '0' if they will have a day off.

At the last moment Gleb said that the camp will be the most productive if it runs with the schedule t (which can be described in the same format as schedule s). Since the number of days in the current may be different from number of days in schedule t, Gleb required that the camp's schedule must be altered so that the number of occurrences of t in it as a substring is maximum possible. At the same time, **the number of contest days and days off shouldn't change**, only their order may change.

Could you rearrange the schedule in the best possible way?

Input

The first line contains string s ($1 \leqslant |s| \leqslant 500\,000$), denoting the current project of the camp's schedule.

The second line contains string t ($1 \leqslant |t| \leqslant 500\,000$), denoting the optimal schedule according to Gleb.

Strings s and t contain characters '0' and '1' only.

Output

In the only line print the schedule having the largest number of substrings equal to t. Printed schedule should consist of characters '0' and '1' only and the number of zeros should be equal to the number of zeros in s and the number of ones should be equal to the number of ones in s.

In case there multiple optimal schedules, print any of them.

Examples

input
```
101101
110
```
output
```
110110
```

input
```
10010110
100011
```
output
```
01100011
```

input
```
10
11100
```
output
```
01
```

Note

In the first example there are two occurrences, one starting from first position and one starting from fourth position.

In the second example there is only one occurrence, which starts from third position. Note, that the answer is not unique. For example, if we move the first day (which is a day off) to the last position, the number of occurrences of t wouldn't change.

In the third example it's impossible to make even a single occurrence.

E. Museums Tour

time limit per test: 4 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

In the country N, there are n cities connected by m one-way roads. Although this country seems unremarkable, there are two interesting facts about it. At first, a week lasts d days here. At second, there is exactly one museum in each city of the country N.

Travel agency "Open museums" is developing a new program for tourists interested in museums. Agency's employees know which days each of the museums is open. The tour should start in the capital — the city number 1, and the first day of the tour must be on the first day of a week. Each day a tourist will be in some city, watching the exposition in its museum (in case museum is open today), and by the end of the day, the tour either ends or the tourist goes into another city connected by a road with the current one. The road system of N is designed in such a way that traveling by a road always takes one night and also all the roads are **one-way**. It's allowed to visit a city multiple times during the trip.

You should develop such route for the trip that the number of **distinct** museums, possible to visit during it, is maximum.

Input

The first line contains three integers n, m and d ($1 \leq n \leq 100\,000$, $0 \leq m \leq 100\,000$, $1 \leq d \leq 50$), the number of cities, the number of roads and the number of days in a week.

Each of next m lines contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$), denoting a **one-way** road from the city u_i to the city v_i.

The next n lines contain the museums' schedule. The schedule of the museum located in the i-th city is described in the i-th of these lines. Each line consists of exactly d characters "`0`" or "`1`", the j-th character of the string equals to "`1`" if the museum is open at the j-th day of a week, and "`0`", otherwise.

It's guaranteed that for each pair of cities (u, v) there exists no more than one road leading from u to v.

Output

Print a single integer — the maximum number of distinct museums, that it's possible to visit, starting a trip in the first city on the first day of the week.

Examples

input

```
4 5 3
3 1
1 2
2 4
4 1
2 3
011
110
111
001
```

output

```
3
```

input

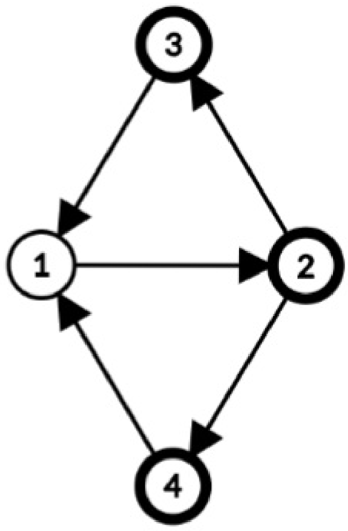
```
3 3 7
1 2
1 3
2 3
1111111
0000000
0111111
```

output

```
2
```

Note

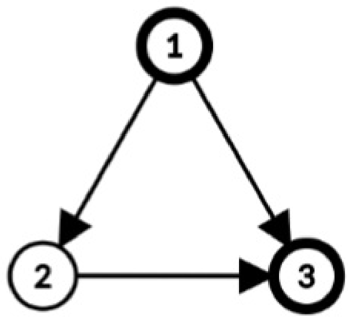
Explanation of the first example



The maximum number of distinct museums to visit is 3. It's possible to visit 3 museums, for example, in the way described below.

- **Day 1.** Now it's the 1st day of a week, and the tourist is in the city 1. The museum there is closed. At night the tourist goes to the city number 2.
- **Day 2.** Now it's the 2nd day of a week, and the tourist is in the city 2. The museum there is open, and the tourist visits it. At night the tourist goes to the city number 4.
- **Day 3.** Now it's the 3rd day of a week, and the tourist is in the city 4. The museum there is open, and the tourist visits it. At night the tourist goes to the city number 1.
- **Day 4.** Now it's the 1st day of a week, and the tourist is in the city 1. The museum there is closed. At night the tourist goes to the city number 2.
- **Day 5.** Now it's the 2nd of a week number 2, and the tourist is in the city 2. The museum there is open, but the tourist has already visited it. At night the tourist goes to the city number 3.
- **Day 6.** Now it's the 3rd day of a week, and the tourist is in the city 3. The museum there is open, and the tourist visits it. After this, the tour is over.

Explanation of the second example



The maximum number of distinct museums to visit is 2. It's possible to visit 2 museums, for example, in the way described below.

- **Day 1.** Now it's the 1st day of a week, and the tourist is in the city 1. The museum there is open, and the tourist visits it. At night the tourist goes to the city number 2.
- **Day 2.** Now it's the 2nd day of a week, and the tourist is in the city 2. The museum there is closed. At night the tourist goes to the city number 3.
- **Day 3.** Now it's the 3rd day of a week, and the tourist is in the city 3. The museum there is open, and the tourist visits it. After this, the tour is over.

F. Cooperative Game

time limit per test: 1 second

memory limit per test: 512 megabytes

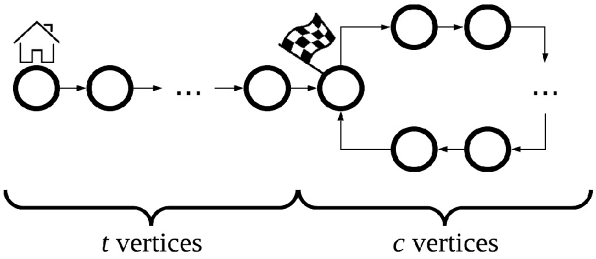
input: standard input

output: standard output

This is an interactive problem.

Misha likes to play cooperative games with incomplete information. Today he suggested ten his friends to play a cooperative game "Lake".

Misha has already come up with a field for the upcoming game. The field for this game is a directed graph consisting of two parts. The first part is a road along the coast of the lake which is a cycle of c vertices. The second part is a path from home to the lake which is a chain of t vertices, and there is an edge from the last vertex of this chain to the vertex of the road along the coast which has the most beautiful view of the lake, also known as the finish vertex. Misha decided to keep the field secret, so nobody knows neither t nor c.



Note that each vertex of the field has exactly one outgoing edge and all the vertices except the home vertex and the finish vertex have exactly one ingoing edge. The home vertex has no incoming edges, the finish vertex has two incoming edges.

At the beginning of the game pieces of all the ten players, indexed with consecutive integers from 0 to 9, are at the home vertex. After that on each turn some of the players can ask Misha to simultaneously move their pieces along the corresponding edges. Misha will not answer more than q such queries. After each move Misha will tell players whose pieces are at the same vertices and whose pieces are at different vertices.

The goal of the game is to move all the pieces to the finish vertex. Misha's friends have no idea how to win in such a game without knowledge of c, t and q, but luckily they are your friends. Help them: coordinate their actions to win the game.

Misha has drawn such a field that $1 \leq t, c, (t+c) \leq 1000$ and $q = 3 \cdot (t+c)$.

Input

There is no input — go to the interaction part straight away.

Output

After all friends gather at the finish vertex, print "`done`" and terminate your program.

Interaction

To give a command to move the friends, print "`next`" and then space-separated indices of the friends you want to move. For example, to give the command to move the friends with indices 0, 2, 5 and 9 print "`next 0 2 5 9`". At each turn, you must move at least one of your friends.

As an answer, first read an integer k, and then 10 digits divided into k space-separated groups. The friends that correspond to the indices in the same group are in the same vertex. The friends that correspond to indices in different groups are in different vertices. The indices in each group follow in ascending order.

For example, the answer "`2 05 12346789`" means that the friends with indices 0 and 5 are in one vertex, and all other friends are in the same but different vertex. The answer "`4 01 567 234 89`" means that Misha's friends are in four different vertices: the friends with indices 0 and 1 are in the first, the friends with indices 5, 6 and 7 are in the second, the friends with indices 2, 3 and 4 are in the third, and the friends with indices 8 and 9 are in the fourth.

After printing a query do not forget to output end of line and flush the output. Otherwise you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Answer "`stop`" instead of a valid one means that you made an invalid query. Exit immediately after receiving "`stop`" and you will see `Wrong answer` verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

Hacks

In order to hack, print two integers t and c in a single line ($1 \leq t, c, (t+c) \leq 1000$).

Example

input
2 05 12346789
3 246789 135 0
3 246789 0 135
3 246789 0 135
2 135 0246789
1 0123456789

output

```
next 0 5

next 0 1 3

next 2 3 0 1 4 5 6 7 8 9

next 9 8 7 6 5 4 3 2 1 0

next 0 1 3 5

next 1 3 5

done
```

Note

In the sample input and output values are aligned only for simplicity of interpreting them chronologically. In real interaction no "extra" line breaks should appear.

In the example, the friends move as follows: