Codeforces Round #559 (Div. 1)

A. The Party and Sweets

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

n boys and m girls came to the party. Each boy presented each girl some integer number of sweets (possibly zero). All boys are numbered with integers from 1 to n and all girls are numbered with integers from 1 to m. For all $1 \leq i \leq n$ the minimal number of sweets, which i-th boy presented to some girl is equal to b_i and for all $1 \leq j \leq m$ the maximal number of sweets, which j-th girl received from some boy is equal to g_j.

More formally, let $a_{i,j}$ be the number of sweets which the i-th boy give to the j-th girl. Then b_i is equal exactly to the minimum among values $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ and g_j is equal exactly to the maximum among values $b_{1,j}, b_{2,j}, \ldots, b_{n,j}$.

You are interested in the minimum total number of sweets that boys could present, so you need to minimize the sum of $a_{i,j}$ for all (i, j) such that $1 \leq i \leq n$ and $1 \leq j \leq m$. You are given the numbers $b_1, \ldots, b_n$ and $g_1, \ldots, g_m$, determine this number.

Input

The first line contains two integers n and m, separated with space — the number of boys and girls, respectively ($2 \leq n, m \leq 100\,000$). The second line contains n integers $b_1, \ldots, b_n$, separated by spaces — b_i is equal to the minimal number of sweets, which i-th boy presented to some girl ($0 \leq b_i \leq 10^8$). The third line contains m integers $g_1, \ldots, g_m$, separated by spaces — g_j is equal to the maximal number of sweets, which j-th girl received from some boy ($0 \leq g_j \leq 10^8$).

Output

If the described situation is impossible, print -1. In another case, print the minimal total number of sweets, which boys could have presented and all conditions could have satisfied.

Examples

input
```
3 2
1 2 1
3 4
```
output
```
12
```

input
```
2 2
0 1
1 0
```
output
```
-1
```

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

Note

In the first test, the minimal total number of sweets, which boys could have presented is equal to 12. This can be possible, for example, if the first boy presented 1 and 4 sweets, the second boy presented 3 and 2 sweets and the third boy presented 1 and 1 sweets for the first and the second girl, respectively. It's easy to see, that all conditions are satisfied and the total number of sweets is equal to 12.

In the second test, the boys couldn't have presented sweets in such way, that all statements satisfied.

In the third test, the minimal total number of sweets, which boys could have presented is equal to 4. This can be possible, for example, if the first boy presented $1, 1, 2$ sweets for the first, second, third girl, respectively and the second boy didn't present sweets for each girl. It's easy to see, that all conditions are satisfied and the total number of sweets is equal to 4.

B. The minimal unique substring

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

Let s be some string consisting of symbols "0" or "1". Let's call a string t a substring of string s, if there exists such number $1 \leq l \leq |s| - |t| + 1$ that $t = s_l s_{l+1} \ldots s_{l+|t|-1}$. Let's call a substring t of string s unique, if there exist only one such l.

For example, let $s =$"1010111". A string $t =$"010" is an unique substring of s, because $l = 2$ is the only one suitable number. But, for example $t =$"10" isn't a unique substring of s, because $l = 1$ and $l = 3$ are suitable. And for example $t =$"00" at all isn't a substring of s, because there is no suitable l.

Today Vasya solved the following problem at the informatics lesson: given a string consisting of symbols "0" and "1", the task is to find the length of its minimal unique substring. He has written a solution to this problem and wants to test it. He is asking you to help him.

You are given 2 positive integers n and k, such that $(n \bmod 2) = (k \bmod 2)$, where $(x \bmod 2)$ is operation of taking remainder of x by dividing on 2. Find any string s consisting of n symbols "0" or "1", such that the length of its minimal unique substring is equal to k.

Input

The first line contains two integers n and k, separated by spaces ($1 \leq k \leq n \leq 100\,000$, $(k \bmod 2) = (n \bmod 2)$).

Output

Print a string s of length n, consisting of symbols "0" and "1". Minimal length of the unique substring of s should be equal to k. You can find **any** suitable string. It is guaranteed, that there exists at least one such string.

Examples

input
```
4 4
```
output
```
1111
```

input
```
5 3
```
output
```
01010
```

input
```
7 3
```
output
```
1011011
```

Note

In the first test, it's easy to see, that the only unique substring of string $s =$"1111" is all string s, which has length 4.

In the second test a string $s =$"01010" has minimal unique substring $t =$"101", which has length 3.

In the third test a string $s =$"1011011" has minimal unique substring $t =$"110", which has length 3.

C. Permutation recovery

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

Vasya has written some permutation $p_1, p_2, \ldots, p_n$ of integers from 1 to n, so for all $1 \leq i \leq n$ it is true that $1 \leq p_i \leq n$ and all $p_1, p_2, \ldots, p_n$ are different. After that he wrote n numbers $next_1, next_2, \ldots, next_n$. The number $next_i$ is equal to the minimal index $i < j \leq n$, such that $p_j > p_i$. If there is no such j let's let's define as $next_i = n + 1$.

In the evening Vasya went home from school and due to rain, his notebook got wet. Now it is impossible to read some written numbers. Permutation and some values $next_i$ are completely lost! If for some i the value $next_i$ is lost, let's say that $next_i = -1$.

You are given numbers $next_1, next_2, \ldots, next_n$ (maybe some of them are equal to -1). Help Vasya to find such permutation $p_1, p_2, \ldots, p_n$ of integers from 1 to n, that he can write it to the notebook and all numbers $next_i$, which are not equal to -1, will be correct.

Input

The first line contains one integer t — the number of test cases ($1 \leq t \leq 100\,000$).

Next $2 \cdot t$ lines contains the description of test cases,two lines for each. The first line contains one integer n — the length of the permutation, written by Vasya ($1 \le n \le 500\,000$). The second line contains n integers $next_1, next_2, \ldots, next_n$, separated by spaces ($next_i = -1$ or $i < next_i \le n + 1$).

It is guaranteed, that the sum of n in all test cases doesn't exceed $500\,000$.

In **hacks** you can only use one test case, so $T = 1$.

Output

Print T lines, in i-th of them answer to the i-th test case.

If there is no such permutations $p_1, p_2, \ldots, p_n$ of integers from 1 to n, that Vasya could write, print the only number -1.

In the other case print n different integers $p_1, p_2, \ldots, p_n$, separated by spaces ($1 \le p_i \le n$). All defined values of $next_i$ which are not equal to -1 should be computed correctly $p_1, p_2, \ldots, p_n$ using defenition given in the statement of the problem. If there exists more than one solution you can find any of them.

Example

input

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

output

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

Note

In the first test case for permutation $p = [1, 2, 3]$ Vasya should write $next = [2, 3, 4]$, because each number in permutation is less than next. It's easy to see, that it is the only satisfying permutation.

In the third test case, any permutation can be the answer because all numbers $next_i$ are lost.

In the fourth test case, there is no satisfying permutation, so the answer is -1.

D. Winding polygonal line

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

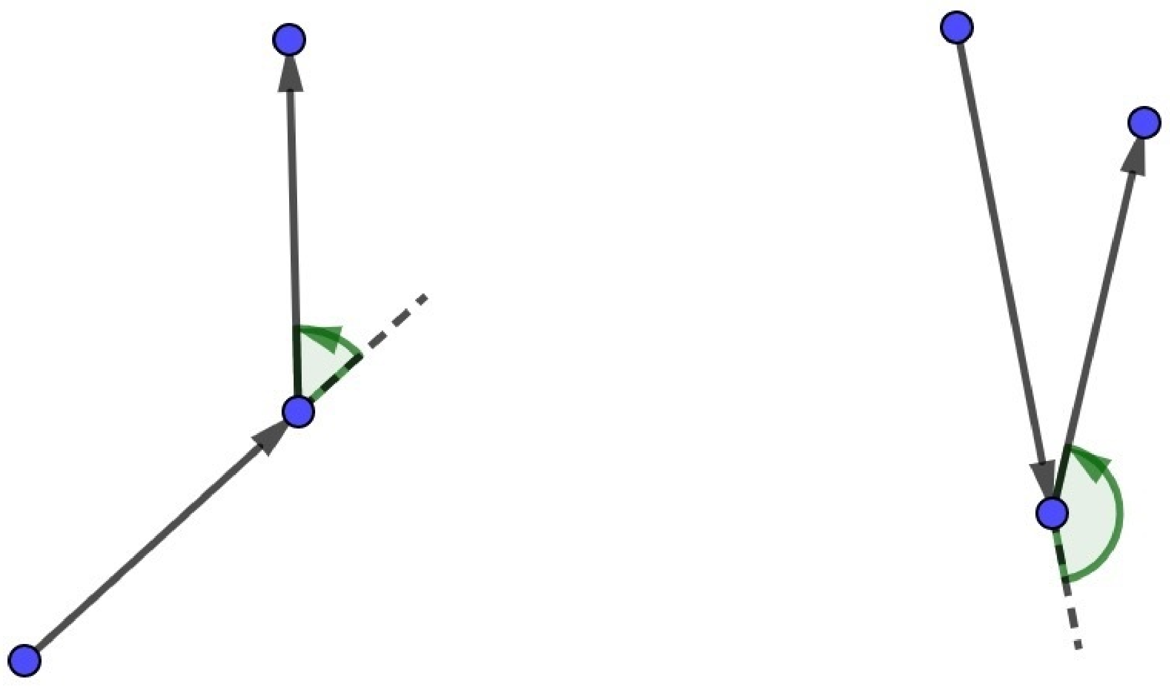
output: standard output

Vasya has n different points $A_1, A_2, \ldots A_n$ on the plane. No three of them lie on the same line He wants to place them in some order $A_{p_1}, A_{p_2}, \ldots, A_{p_n}$, where $p_1, p_2, \ldots, p_n$ — some permutation of integers from 1 to n.

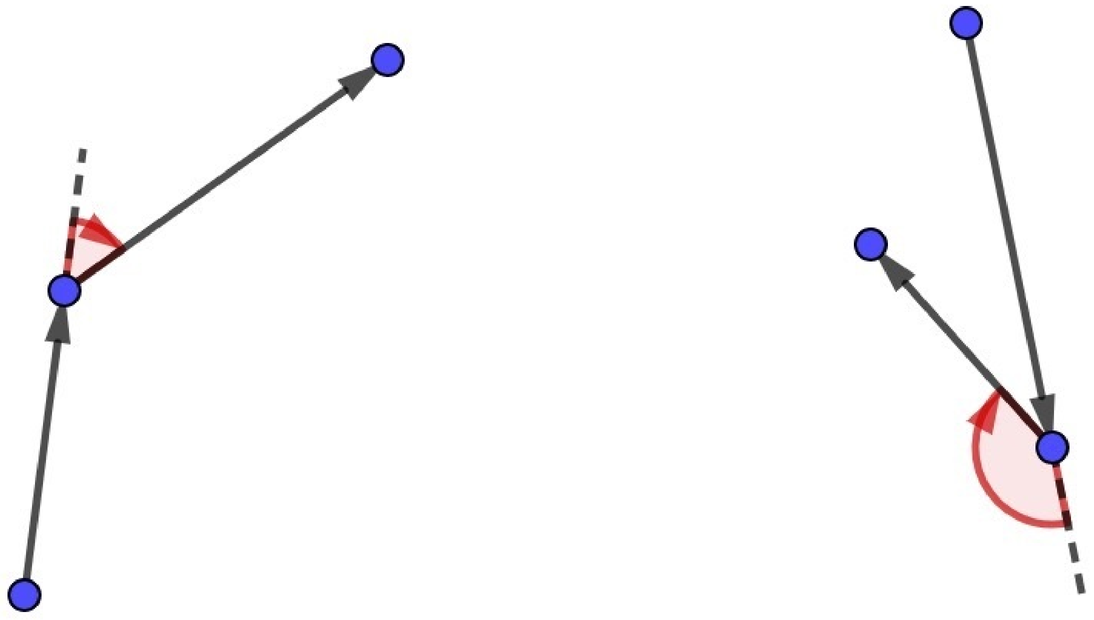
After doing so, he will draw oriented polygonal line on these points, drawing oriented segments from each point to the next in the chosen order. So, for all $1 \le i \le n-1$ he will draw oriented segment from point A_{p_i} to point $A_{p_{i+1}}$. He wants to make this polygonal line satisfying 2 conditions:

- it will be non-self-intersecting, so any 2 segments which are not neighbors don't have common points.
- it will be **winding**.

Vasya has a string s, consisting of $(n-2)$ symbols "L" or "R". Let's call an oriented polygonal line **winding**, if its i-th turn left, if $s_i =$ "L" and right, if $s_i =$ "R". More formally: i-th turn will be in point $A_{p_{i+1}}$, where oriented segment from point A_{p_i} to point $A_{p_{i+1}}$ changes to oriented segment from point $A_{p_{i+1}}$ to point $A_{p_{i+2}}$. Let's define vectors $\overrightarrow{v_1} = \overrightarrow{A_{p_i} A_{p_{i+1}}}$ and $\overrightarrow{v_2} = \overrightarrow{A_{p_{i+1}} A_{p_{i+2}}}$. Then if in order to rotate the vector $\overrightarrow{v_1}$ by the smallest possible angle, so that its direction coincides with the direction of the vector $\overrightarrow{v_2}$ we need to make a turn counterclockwise, then we say that i-th turn is to the left, and otherwise to the right. For better understanding look at this pictures with some examples of turns:



There are left turns on this picture



There are right turns on this picture

You are given coordinates of the points $A_1, A_2, \dots A_n$ on the plane and string s. Find a permutation $p_1, p_2, \dots, p_n$ of the integers from 1 to n, such that the polygonal line, drawn by Vasya satisfy two necessary conditions.

Input

The first line contains one integer n — the number of points ($3 \le n \le 2000$). Next n lines contains two integers x_i and y_i, divided by space — coordinates of the point A_i on the plane ($-10^9 \le x_i, y_i \le 10^9$). The last line contains a string s consisting of symbols "L" and "R" with length $(n-2)$. It is guaranteed that all points are different and no three points lie at the same line.

Output

If the satisfying permutation doesn't exists, print -1. In the other case, print n numbers $p_1, p_2, \dots, p_n$ — the permutation which was found ($1 \le p_i \le n$ and all $p_1, p_2, \dots, p_n$ are different). If there exists more than one solution, you can find any.

Examples

input

```
3
1 1
3 1
1 3
L
```

output

```
1 2 3
```

input

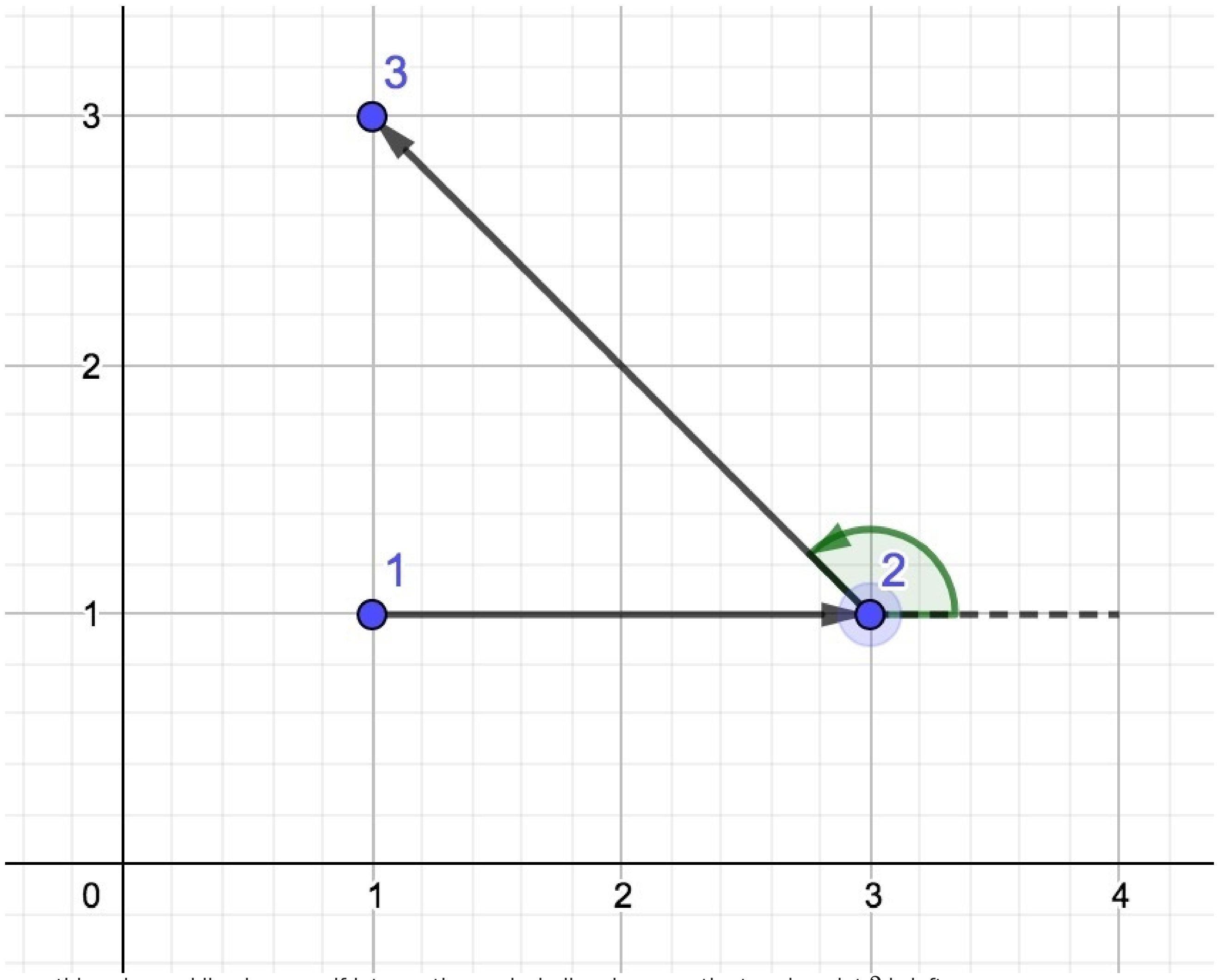
```
6
1 0
0 1
0 2
-1 0
-1 -1
2 1
RLLR
```

output

```
6 1 3 4 2 5
```

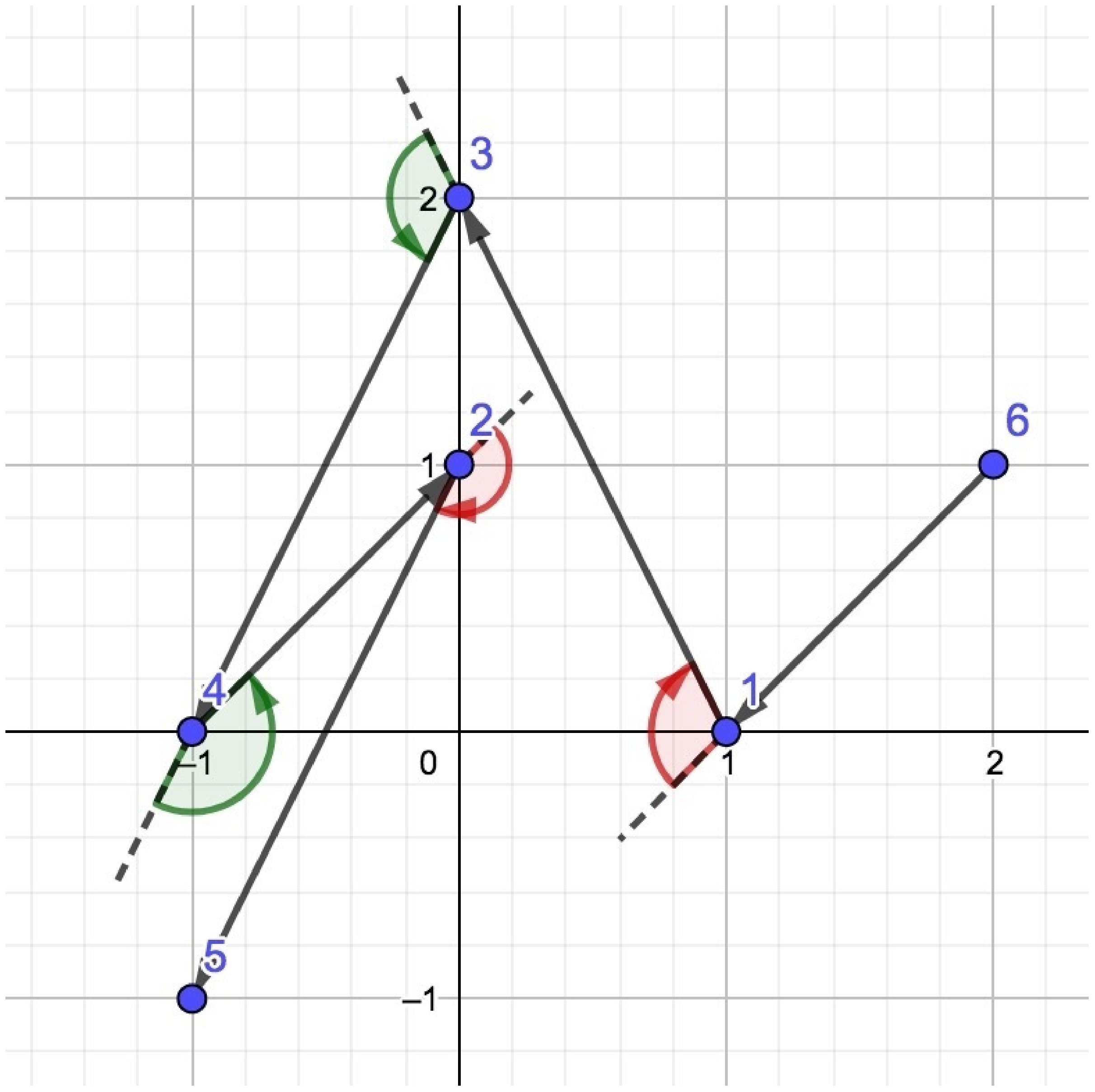
Note

This is the picture with the polygonal line from the 1 test:



As we see, this polygonal line is non-self-intersecting and winding, because the turn in point 2 is left.

This is the picture with the polygonal line from the 2 test:



E. Strange device

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

It is an interactive problem.

Vasya enjoys solving quizzes. He found a strange device and wants to know how it works.

This device encrypted with the tree (connected undirected graph without cycles) with n vertices, numbered with integers from 1 to n. To solve this quiz you should guess this tree.

Fortunately, this device can make one operation, using which you should guess the cipher. You can give the device an array $d_1, d_2, \ldots, d_n$ of non-negative integers. On the device, there are n lamps, i-th of them is connected with i-th vertex of the tree. For all i the light will turn on the i-th lamp, if there exist such vertex of the tree with number $j \neq i$ that $dist(i, j) \leq d_j$. Let's define $dist(i, j)$ as the distance between vertices i and j in tree or number of edges on the simple path between vertices i and j.

Vasya wants to solve this quiz using ≤ 80 operations with the device and guess the tree. Help him!

Interaction

In the beginning, your program should read one integer n — the number of vertices of the tree which encrypts the device ($2 \leq n \leq 1000$).

After that, you can make several operations in the following format. To do operation print a symbol"?" (without quotes) and n integers $d_1, d_2, \ldots, d_n$, separated by spaces after it. Please note, that for all i you can only use the numbers, satisfying the inequality $0 \leq d_i < n$. After that, you should read a string s with length n, consisting of symbols "0" and "1" (without quotes). For all i the symbol s_i is equal to "0", if the lamp on the device, connected with i-th vertex of the tree is switched off and "1" otherwise.

After several operations, you should print guessed tree. To do it print the only symbol "!" (without quotes). In the next $n-1$ lines print 2 integers a_i, b_i — indexes of the vertices connected by i-th edge of the tree. This numbers should satisfy the conditions $1 \leq a_i, b_i \leq n$ and $a_i \neq b_i$. This edges should form a tree, which is equal to the hidden tree. After that, your program should terminate.

It is guaranteed, that in each test the tree is fixed before and won't change depending on your program's operations.

Your program can make from 0 to 80 operations with the device and after that guess the tree equal with the hidden.

If your program will make more than 80 operations it can get any verdict, because it will continue reading from closed input. If your program will make operation or print the answer in the incorrect format, it can get any verdict too. **Be careful.**

Don't forget to flush the output after printing questions and answers.

To flush the output, you can use:

- fflush(stdout) in C++.
- System.out.flush() in Java.
- stdout.flush() in Python.
- flush(output) in Pascal.
- See the documentation for other languages.

Hacks:

The first line should contain one integer n — the number of vertices in the tree ($2 \leq n \leq 1000$). Next $n-1$ lines should contain 2 integers a_i, b_i — indexes of the vertices connected by i-th edge of the tree ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$). All edges should form a tree. Be careful, extra spaces or line breaks are not allowed.

Example

input

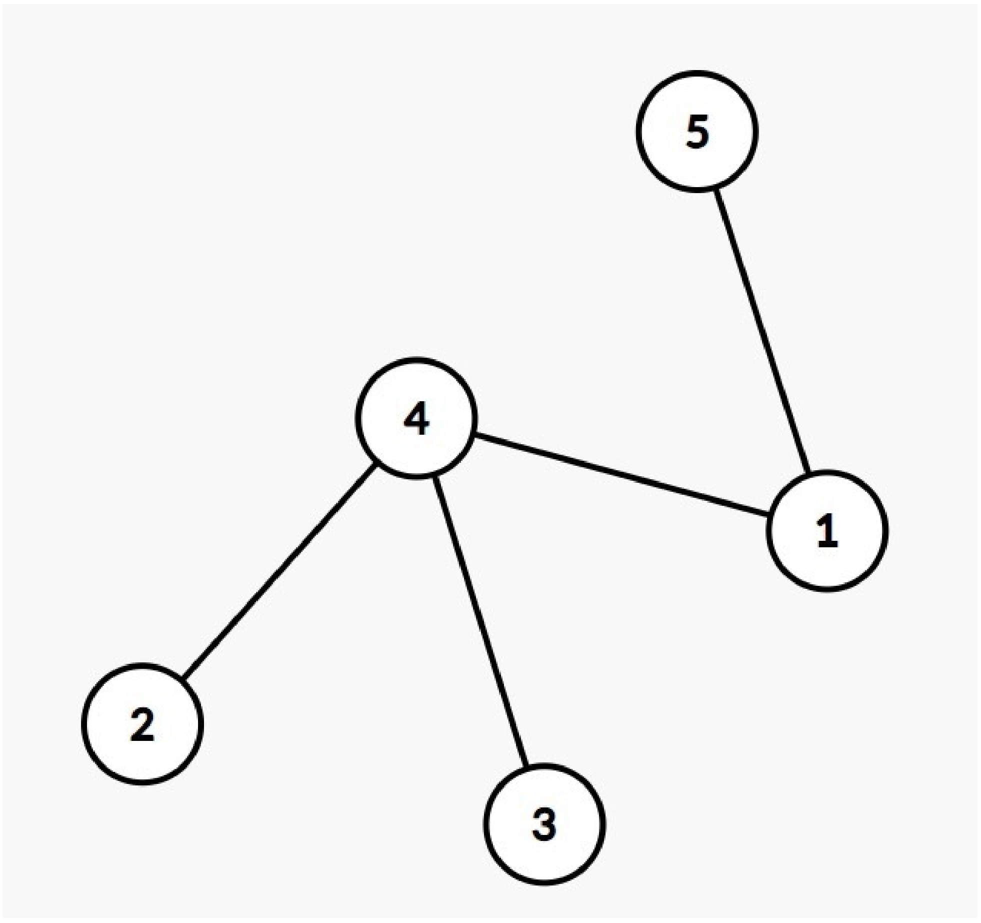
```
5
00000
11011
11100
10010
```

output

```
? 0 0 0 0 0
? 1 1 2 0 2
? 0 0 0 1 0
? 0 1 0 0 1
!
4 2
1 5
3 4
4 1
```

Note

It is a picture of the tree which encrypt the device from the first test:



It is a table of pairwise distances between vertices in this tree:

	1	2	3	4	5
1	0	2	2	1	1
2	2	0	2	1	3
3	2	2	0	1	3
4	1	1	1	0	2
5	1	3	3	2	0

- If you make operation where $d = [0, 0, 0, 0, 0]$, no lamp will switch on, because $dist(i, j) > 0$ for all $i \neq j$.
- If you make operation where $d = [1, 1, 2, 0, 2]$, all lamps except the lamp connected with the 3-rd vertex will switch on. For example, lamp connected with the 1-st vertex will switch on, because $dist(1, 5) = 1 \leq 2 = d_5$.
- If you make operation where $d = [0, 0, 0, 1, 0]$, all lamps except lamps connected with the 4-th and 5-th vertices will switch on.
- If you make operation where $d = [0, 1, 0, 0, 1]$, only lamps connected with the 1-st and 4-th vertices will switch on.

F. Density of subarrays

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let c be some positive integer. Let's call an array $a_1, a_2, \ldots, a_n$ of positive integers c-array, if for all i condition $1 \leq a_i \leq c$ is satisfied. Let's call c-array $b_1, b_2, \ldots, b_k$ a **subarray** of c-array $a_1, a_2, \ldots, a_n$, if there exists such set of k indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ that $b_j = a_{i_j}$ for all $1 \leq j \leq k$. Let's define **density** of c-array $a_1, a_2, \ldots, a_n$ as maximal non-negative integer p, such that any c-array, that contains p numbers is a subarray of $a_1, a_2, \ldots, a_n$.

You are given a number c and some c-array $a_1, a_2, \ldots, a_n$. For all $0 \leq p \leq n$ find the number of sequences of indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ for all $1 \leq k \leq n$, such that density of array $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ is equal to p. Find these numbers by modulo $998\,244\,353$, because they can be too large.

Input

The first line contains two integers n and c, separated by spaces ($1 \leq n, c \leq 3\,000$). The second line contains n integers $a_1, a_2, \ldots, a_n$, separated by spaces ($1 \leq a_i \leq c$).

Output

Print $n + 1$ numbers $s_0, s_1, \ldots, s_n$. s_p should be equal to the number of sequences of indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ for all $1 \leq k \leq n$ by modulo $998\,244\,353$, such that the density of array $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ is equal to p.

Examples

input

```
4 1
1 1 1 1
```

output

```
0 4 6 4 1
```

input

```
3 3
1 2 3
```

output

```
6 1 0 0
```

input

```
5 2
1 2 1 2 1
```

output

```
10 17 4 0 0 0
```

Note

In the first example, it's easy to see that the density of array will always be equal to its length. There exists 4 sequences with one index, 6 with two indices, 4 with three and 1 with four.

In the second example, the only sequence of indices, such that the array will have non-zero density is all indices because in other cases there won't be at least one number from 1 to 3 in the array, so it won't satisfy the condition of density for $p \geq 1$.

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform