A. Forgetting Things

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

Kolya is very absent-minded. Today his math teacher asked him to solve a simple problem with the equation $a+1=b$ with positive integers a and b, but Kolya forgot the numbers a and b. He does, however, remember that the first (leftmost) digit of a was d_a, and the first (leftmost) digit of b was d_b.

Can you reconstruct any equation $a+1=b$ that satisfies this property? It may be possible that Kolya misremembers the digits, and there is no suitable equation, in which case report so.

Input

The only line contains two space-separated digits d_a and d_b ($1 \leq d_a, d_b \leq 9$).

Output

If there is no equation $a+1=b$ with positive integers a and b such that the first digit of a is d_a, and the first digit of b is d_b, print a single number -1.

Otherwise, print any suitable a and b that **both** are positive and do not exceed 10^9. It is guaranteed that if a solution exists, there also exists a solution with both numbers not exceeding 10^9.

Examples

input
```
1 2
```
output
```
199 200
```

input
```
4 4
```
output
```
412 413
```

input
```
5 7
```
output
```
-1
```

input
```
6 2
```
output
```
-1
```

B1. TV Subscriptions (Easy Version)

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

The only difference between easy and hard versions is constraints.

The BerTV channel every day broadcasts one episode of one of the k TV shows. You know the schedule for the next n days: a sequence of integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le k$), where a_i is the show, the episode of which will be shown in i-th day.

The subscription to the show is bought for the entire show (i.e. for all its episodes), for each show the subscription is bought separately.

How many minimum subscriptions do you need to buy in order to have the opportunity to watch episodes of purchased shows d ($1 \le d \le n$) days in a row? In other words, you want to buy the minimum number of TV shows so that there is some segment of d

consecutive days in which all episodes belong to the purchased shows.

Input

The first line contains an integer t ($1 \le t \le 100$) — the number of test cases in the input. Then t test case descriptions follow.

The first line of each test case contains three integers n, k and d ($1 \le n \le 100$, $1 \le k \le 100$, $1 \le d \le n$). The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le k$), where a_i is the show that is broadcasted on the i-th day.

It is guaranteed that the sum of the values of n for all test cases in the input does not exceed 100.

Output

Print t integers — the answers to the test cases in the input in the order they follow. The answer to a test case is the minimum number of TV shows for which you need to purchase a subscription so that you can watch episodes of the purchased TV shows on BerTV for d consecutive days. Please note that it is permissible that you will be able to watch more than d days in a row.

Example

input

```
4
5 2 2
1 2 1 2 1
9 3 3
3 3 3 2 2 2 1 1 1
4 10 4
10 8 6 4
16 9 8
3 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3
```

output

```
2
1
4
5
```

Note

In the first test case to have an opportunity to watch shows for two consecutive days, you need to buy a subscription on show 1 and on show 2. So the answer is two.

In the second test case, you can buy a subscription to any show because for each show you can find a segment of three consecutive days, consisting only of episodes of this show.

In the third test case in the unique segment of four days, you have four different shows, so you need to buy a subscription to all these four shows.

In the fourth test case, you can buy subscriptions to shows $3, 5, 7, 8, 9$, and you will be able to watch shows for the last eight days.

B2. TV Subscriptions (Hard Version)

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

The only difference between easy and hard versions is constraints.

The BerTV channel every day broadcasts one episode of one of the k TV shows. You know the schedule for the next n days: a sequence of integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le k$), where a_i is the show, the episode of which will be shown in i-th day.

The subscription to the show is bought for the entire show (i.e. for all its episodes), for each show the subscription is bought separately.

How many minimum subscriptions do you need to buy in order to have the opportunity to watch episodes of purchased shows d ($1 \le d \le n$) days in a row? In other words, you want to buy the minimum number of TV shows so that there is some segment of d consecutive days in which all episodes belong to the purchased shows.

Input

The first line contains an integer t ($1 \le t \le 10000$) — the number of test cases in the input. Then t test case descriptions follow.

The first line of each test case contains three integers n, k and d ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le 10^6$, $1 \le d \le n$). The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le k$), where a_i is the show that is broadcasted on the i-th day.

It is guaranteed that the sum of the values of n for all test cases in the input does not exceed $2 \cdot 10^5$.

Output

Print t integers — the answers to the test cases in the input in the order they follow. The answer to a test case is the minimum number of TV shows for which you need to purchase a subscription so that you can watch episodes of the purchased TV shows on BerTV for d consecutive days. Please note that it is permissible that you will be able to watch more than d days in a row.

Example

input

```
4
5 2 2
1 2 1 2 1
9 3 3
3 3 3 2 2 2 1 1 1
4 10 4
10 8 6 4
16 9 8
3 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3
```

output

```
2
1
4
5
```

Note

In the first test case to have an opportunity to watch shows for two consecutive days, you need to buy a subscription on show 1 and on show 2. So the answer is two.

In the second test case, you can buy a subscription to any show because for each show you can find a segment of three consecutive days, consisting only of episodes of this show.

In the third test case in the unique segment of four days, you have four different shows, so you need to buy a subscription to all these four shows.

In the fourth test case, you can buy subscriptions to shows $3, 5, 7, 8, 9$, and you will be able to watch shows for the last eight days.

C. p-binary

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

Vasya will fancy any number as long as it is an integer power of two. Petya, on the other hand, is very conservative and only likes a single integer p (which may be positive, negative, or zero). To combine their tastes, they invented *p-binary numbers* of the form $2^x + p$, where x is a **non-negative** integer.

For example, some -9-binary ("minus nine" binary) numbers are: -8 (minus eight), 7 and 1015 ($-8 = 2^0 - 9$, $7 = 2^4 - 9$, $1015 = 2^{10} - 9$).

The boys now use p-binary numbers to represent everything. They now face a problem: given a positive integer n, what's the smallest number of p-binary numbers (not necessarily distinct) they need to represent n as their sum? It may be possible that representation is impossible altogether. Help them solve this problem.

For example, if $p = 0$ we can represent 7 as $2^0 + 2^1 + 2^2$.

And if $p = -9$ we can represent 7 as one number $(2^4 - 9)$.

Note that negative p-binary numbers are allowed to be in the sum (see the Notes section for an example).

Input

The only line contains two integers n and p ($1 \leq n \leq 10^9$, $-1000 \leq p \leq 1000$).

Output

If it is impossible to represent n as the sum of any number of p-binary numbers, print a single integer -1. Otherwise, print the smallest possible number of summands.

Examples

input

```
24 0
```

output

```
2
```

input

```
24 1
```

output

```
3
```

input

```
24 -1
```

output

```
4
```

input
4 -7
output
2

input
1 1
output
-1

Note

0-binary numbers are just regular binary powers, thus in the first sample case we can represent $24 = (2^4 + 0) + (2^3 + 0)$.

In the second sample case, we can represent $24 = (2^4 + 1) + (2^2 + 1) + (2^0 + 1)$.

In the third sample case, we can represent $24 = (2^4 - 1) + (2^2 - 1) + (2^2 - 1) + (2^2 - 1)$. Note that repeated summands are allowed.

In the fourth sample case, we can represent $4 = (2^4 - 7) + (2^1 - 7)$. Note that the second summand is negative, which is allowed.

In the fifth sample case, no representation is possible.

D. Power Products

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

You are given n positive integers $a_1, \ldots, a_n$, and an integer $k \geq 2$. Count the number of pairs i, j such that $1 \leq i < j \leq n$, and there exists an integer x such that $a_i \cdot a_j = x^k$.

Input

The first line contains two integers n and k ($2 \leq n \leq 10^5$, $2 \leq k \leq 100$).

The second line contains n integers $a_1, \ldots, a_n$ ($1 \leq a_i \leq 10^5$).

Output

Print a single integer — the number of suitable pairs.

Example

input
6 3 1 3 9 8 24 1
output
5

Note

In the sample case, the suitable pairs are:

- $a_1 \cdot a_4 = 8 = 2^3$;
- $a_1 \cdot a_6 = 1 = 1^3$;
- $a_2 \cdot a_3 = 27 = 3^3$;
- $a_3 \cdot a_5 = 216 = 6^3$;
- $a_4 \cdot a_6 = 8 = 2^3$.

E. Rock Is Push

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

You are at the top left cell $(1, 1)$ of an $n \times m$ labyrinth. Your goal is to get to the bottom right cell (n, m). You can only move right or down, one cell per step. Moving right from a cell (x, y) takes you to the cell $(x, y + 1)$, while moving down takes you to the cell $(x + 1, y)$.

Some cells of the labyrinth contain rocks. When you move to a cell with rock, the rock is pushed to the next cell in the direction you're moving. If the next cell contains a rock, it gets pushed further, and so on.

The labyrinth is surrounded by impenetrable walls, thus any move that would put you or any rock outside of the labyrinth is illegal.

Count the number of different legal paths you can take from the start to the goal modulo $10^9 + 7$. Two paths are considered different if there is at least one cell that is visited in one path, but not visited in the other.

Input

The first line contains two integers n, m — dimensions of the labyrinth ($1 \le n, m \le 2000$).

Next n lines describe the labyrinth. Each of these lines contains m characters. The j-th character of the i-th of these lines is equal to "R" if the cell (i, j) contains a rock, or "." if the cell (i, j) is empty.

It is guaranteed that the starting cell $(1, 1)$ is empty.

Output

Print a single integer — the number of different legal paths from $(1, 1)$ to (n, m) modulo $10^9 + 7$.

Examples

input

```
1 1
.
```

output

```
1
```

input

```
2 3
...
..R
```

output

```
0
```

input

```
4 4
...R
.RR.
.RR.
R...
```

output

```
4
```

Note

In the first sample case we can't (and don't have to) move, hence the only path consists of a single cell $(1, 1)$.

In the second sample case the goal is blocked and is unreachable.

Illustrations for the third sample case can be found here: https://assets.codeforces.com/rounds/1225/index.html

F. Tree Factory

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

Bytelandian Tree Factory produces trees for all kinds of industrial applications. You have been tasked with optimizing the production of a certain type of tree for an especially large and important order.

The tree in question is a rooted tree with n vertices labelled with distinct integers from 0 to $n - 1$. The vertex labelled 0 is the root of the tree, and for any non-root vertex v the label of its parent $p(v)$ is less than the label of v.

All trees at the factory are made from bamboo blanks. A *bamboo* is a rooted tree such that each vertex has exactly one child, except for a single leaf vertex with no children. The vertices of a bamboo blank can be labelled arbitrarily before its processing is started.

To process a bamboo into another tree a single type of operation can be made: choose an arbitrary non-root vertex v such that its parent $p(v)$ is not a root either. The operation consists of changing the parent of v to its parent's parent $p(p(v))$. Note that parents of all other vertices remain unchanged, in particular, the subtree of v does not change.

Efficiency is crucial, hence you have to minimize the number of operations to make the desired tree from a bamboo blank. Construct any optimal sequence of operations to produce the desired tree.

Note that the labelling of the resulting tree has to coincide with the labelling of the desired tree. Formally, the labels of the roots have to be equal, and for non-root vertices with the same label the labels of their parents should be the same.

It is guaranteed that for any test present in this problem an answer exists, and further, an optimal sequence contains at most 10^6 operations. Note that **any hack that does not meet these conditions will be invalid**.

Input

The first line contains a single integer n — the number of vertices in the tree ($2 \le n \le 10^5$).

The second line contains $n-1$ integers $p(1), \ldots, p(n-1)$ — indices of parent vertices of $1, \ldots, n-1$ respectively ($0 \le p(i) < i$).

Output

In the first line, print n distinct integers $id_1, \ldots, id_n$ — the initial labelling of the bamboo blank starting from the root vertex ($0 \le id_i < n$).

In the second line, print a single integer k — the number of operations in your sequence ($0 \le k \le 10^6$).

In the third line print k integers $v_1, \ldots, v_k$ describing operations in order. The i-th operation consists of changing $p(v_i)$ to $p(p(v_i))$. Each operation should be valid, i.e. neither v_i nor $p(v_i)$ can be the root of the tree at the moment.

Examples

input

```
5
0 0 1 1
```

output

```
0 2 1 4 3
2
1 3
```

input

```
4
0 1 2
```

output

```
0 1 2 3
0
```

G. To Make 1

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

There are n positive integers written on the blackboard. Also, a positive number $k \ge 2$ is chosen, and none of the numbers on the blackboard are divisible by k. In one operation, you can choose any two integers x and y, erase them and write one extra number $f(x+y)$, where $f(x)$ is equal to x if x is not divisible by k, otherwise $f(x) = f(x/k)$.

In the end, there will be a single number of the blackboard. Is it possible to make the final number equal to 1? If so, restore any sequence of operations to do so.

Input

The first line contains two integers n and k — the initial number of integers on the blackboard, and the chosen number ($2 \le n \le 16$, $2 \le k \le 2000$).

The second line contains n positive integers $a_1, \ldots, a_n$ initially written on the blackboard. It is guaranteed that none of the numbers a_i is divisible by k, and the sum of all a_i does not exceed 2000.

Output

If it is impossible to obtain 1 as the final number, print "NO" in the only line.

Otherwise, print "YES" on the first line, followed by $n-1$ lines describing operations. The i-th of these lines has to contain two integers x_i and y_i to be erased and replaced with $f(x_i + y_i)$ on the i-th operation. If there are several suitable ways, output any of them.

Examples

input

```
2 2
1 1
```

output

```
YES
1 1
```

input

```
4 3
7 8 13 23
```

output

```
YES
23 13
8 7
5 4
```

input

```
3 4
1 2 3
```

output

```
NO
```

Note

In the second sample case:

- $f(8+7) = f(15) = f(5) = 5$;
- $f(23+13) = f(36) = f(12) = f(4) = 4$;
- $f(5+4) = f(9) = f(3) = f(1) = 1$.