Problem A. The Best Problem of 2021

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

The year is 2021. People still care about COVID, NNSU just won ICPC 2020, and he is already crazy, we just don't know yet how much. We are looking for the problems for the SnackDown finals and **7dan** suggested this one. For some reason, we decided not to use it then, but internally it became known as *The Best Problem of 2021*.

You are given an array B of numbers and a number X. Calculate (modulo 998 244 353, obviously) the number of subsets S of $\{1, 2, \ldots, X\}$ such that B is one of its bases if we consider the numbers to be vectors over $\mathbf{Z}_2$ with bitwise XOR as vector addition. B is considered to be a basis of S if it is an array of minimum size such that every element of S can be written as bitwise XOR of elements of B.

Input

The first line contains two integers n and m ($1 \le n, m \le 2000$) — the size of B and the length of our numbers in binary. All elements of B and the number X will be given in their binary representation with a length of exactly m (possibly with leading zeroes).

Each of the next n lines contains a binary string of length m which represents an element of B.

The last line contains a binary string of length m which represents the number X.

Output

You'll figure it out.

Examples

standard input	*standard output*
4 4 0001 0010 0100 1000 1101	7364
3 2 00 00 00 11	0
2 3 110 101 101	1
3 10 1111100110 0011110100 0101100001 1110000001	38

Problem B. Random Interactive Convex Hull Bot

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

How do setters come up with problems? Sometimes they just take a couple of buzzwords and smash them together. But we are in 2023, so this totally can be outsourced to AI. Introducing our creation based on ChatGPT — RICH B! And its first official problem:

Prompt: Random Interactive Convex Hull

Problem: A set of n points is chosen uniformly at random among all sets of 2D points with positive integer coordinates up to 10^9 of size n so that no three points lie on the same line. Your task is to find their convex hull. But you are not given the points. Instead, you can make queries of the form "`? i j k`", and the jury program will respond to you with 1 if the turn from $\overrightarrow{P_iP_j}$ to $\overrightarrow{P_iP_k}$ is counter-clockwise, and it will respond with -1 if the turn is clockwise. You can interpret it as $\mathbf{sgn}\left(\overrightarrow{P_iP_j} \times \overrightarrow{P_iP_k}\right)$, where $\times$ is cross product. When you think that you know the convex hull, print it as "`! k i1 i2 ... ik`", where k is the size of the convex hull and $i_1, i_2, \ldots, i_k$ are the indices of points on the convex hull in counter-clockwise order. Any point can be the first one. Constraints: $3 \le n \le 5000$ and you can make at most 30 000 queries.

Interaction Protocol

Read n ($3 \le n \le 5000$).

Then start asking queries by printing "`? i j k`" ($1 \le i, j, k \le n$, $\{i, j, k\}$ are distinct). After each query read the response, which is either 1 or -1. **You can make at most 30 000 queries.**

Don't forget to flush the output, you are not a baby, you know how to do this. Don't do invalid queries, that might cause weird verdicts and you don't want that.

After making all the queries you want, print the answer as "`! k i1 i2 ... ik`", where k is the size of the convex hull and $i_1, i_2, \ldots, i_k$ are the indices of points on the convex hull in counter-clockwise order. This is not counted as a query.

It is guaranteed that the set of points is chosen uniformly at random among all sets of 2D points with positive integer coordinates up to 10^9 of size n such that no three points lie on the same line. The order of the points is also uniformly random. The interactor is **not adaptive**.

Example

standard input	*standard output*
5 1 -1 1 -1 -1 -1 1	 ? 3 1 4 ? 3 5 1 ? 3 4 5 ? 1 5 4 ? 2 3 1 ? 2 4 3 ? 2 4 5 ! 4 1 4 5 2

Problem C. Record Parity

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are given a permutation of length n and an integer k.

An element is called a record if it is strictly greater than all the elements before it.

Calculate the sum of $(-1)^{len}$ over all subsequences that have exactly k records. Here *len* is the number of elements in the subsequence. Since the answer can be large, calculate it modulo 998 244 353.

Input

The first line contains two integers n and k ($1 \le k \le n \le 10^6$).

The second line contains the permutation $p_1, p_2, \ldots, p_n$.

Output

I'll let you guess this one.

Examples

standard input	*standard output*
5 2 4 1 2 5 3	3
7 3 1 2 3 4 5 6 7	998244318
5 5 2 5 4 1 3	0

Note

In the second sample all of subsequences of length 3 have exactly 3 records, and none other subsequences have exactly 3 records, so the sum is equal to $(-1)^3\binom{7}{3} = -35$, which is 998 244 318 modulo 998 244 353.

In the third sample none of the subsequences have exactly 5 records, and the sum of empty set is 0.

Problem D. XOR Determinant

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are given two arrays b and c of length n, consisting of non-negative integers. Construct $n \times n$ matrix A as $A_{ij} = b_i \oplus c_j$. Find the determinant of A modulo 998 244 353.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases. The descriptions of the t test cases follow.

The first line of each test case contains one integer n ($1 \le n \le 5000$).

The second line contains the array $b_1, b_2, \ldots, b_n$ ($0 \le b_i < 2^{60}$).

The third line contains the array $c_1, c_2, \ldots, c_n$ ($0 \le c_i < 2^{60}$).

The sum of n over all test cases does not exceed 10 000.

Output

For each test case, print the determinant of matrix A modulo 998 244 353.

Example

standard input	*standard output*
3 2 2 5 4 1 1 1000000000000000001 987467354324283836 4 1 2 3 4 1 2 3 4	21 214139910 998244129

Note

First test case:

$$\begin{vmatrix} 6 & 3 \\ 1 & 4 \end{vmatrix} = 6 \cdot 4 - 1 \cdot 3 = 21$$

Second test case:

$|23\,792\,195\,055\,071\,677| = 23\,792\,195\,055\,071\,677$

$23\,792\,195\,055\,071\,677 \bmod 998\,244\,353 = 214\,139\,910$

Third test case:

$$\begin{vmatrix} 0 & 3 & 2 & 5 \\ 3 & 0 & 1 & 6 \\ 2 & 1 & 0 & 7 \\ 5 & 6 & 7 & 0 \end{vmatrix} = 3 \cdot 3 \cdot 7 \cdot 7 - 3 \cdot 1 \cdot 7 \cdot 5 - 3 \cdot 6 \cdot 2 \cdot 7 - 2 \cdot 3 \cdot 7 \cdot 6 + 2 \cdot 6 \cdot 2 \cdot 6 - 2 \cdot 6 \cdot 1 \cdot 5 - 5 \cdot 3 \cdot 1 \cdot 7 - 5 \cdot 1 \cdot 2 \cdot 6 + 5 \cdot 1 \cdot 1 \cdot 5 =$$

$= 441 - 105 - 252 - 252 + 144 - 60 - 105 - 60 + 25 = -224$

$(-224) \bmod 998\,244\,353 = 998\,244\,129$

Problem E. Egor Has a Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Egor has come up with a hard problem for a training camp! Here it is:

Given an array a of n positive integers sorted in increasing order, find 4 indices $i < j < p < q$ such that $a_i \cdot a_q = a_j \cdot a_p$.

He then wrote the checker to this problem:

```
// returns true if the solution is found,
// returns false if the solution is not found,
// makes the verdict Wrong Answer right away if the found solution is not valid
bool getAnswer(InStream &stream, vector<long long> a) {
    string s = stream.readToken("NO|YES"); // PE if the string is not NO or YES
    if (s == "NO") return false;
    vector<int> b = stream.readInts(4, 1, (int)a.size()); // 4 indices between 1 and n
    int i = b[0] - 1, j = b[1] - 1, p = b[2] - 1, q = b[3] - 1;  // -1 to make 0-indexed
    stream.ensuref(i < j && j < p && p < q, "4 indices should be in increasing order");
    stream.ensuref(a[q] / a[p] == a[j] / a[i], "the products are not equal");
    return true;
}
```

The multiplication will overflow `long long`, so Egor used division instead. How smart! Although now Egor might have another problem...

Input

The first line contains one integer n ($4 \le n \le 500\,000$) — the size of the array.

The second line contains the array $a_1, a_2, \ldots, a_n$ itself ($1 \le a_1 < a_2 < \ldots < a_n \le 10^{18}$).

Output

On the first line print "`YES`" if there is a solution and print "`NO`" otherwise.

If a solution exists, print the 4 chosen indices in order i, j, p, q, separated by spaces. If there is more than one solution, you can print any one.

Examples

standard input	*standard output*
6 2 6 11 21 47 120	YES 1 3 4 6
5 1 2 6 30 210	NO
4 7 13 77 143	YES 1 2 3 4
4 10 29 31 100	NO

Note

The code in the statement is a snippet from the actual checker for **this** problem. Here is the link to the full code with highlighting: `https://pastebin.com/3ZpNUA6f`, password: "`gkVcB4iqwE`".

Problem F. Is This FFT?

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15 seconds
Memory limit:	998 244 353 bytes

Consider the following method of generating a random tree of size n:

Take all $n(n-1)/2$ possible edges, and choose a uniformly random permutation on them. Start with an empty graph, iterate over the edges in the chosen order, and if the edge connects two different connected components, add it to the graph. In the end, you will get a tree.

For all n from 2 to given N, calculate the probability that this algorithm generates a bamboo modulo given P. It is guaranteed that P is prime and $P \bmod 2^{16} = 1$.

Recall that bamboo is a tree in which degrees of all vertices are at most 2.

Since we give you the modulus, you might want to use Barrett reduction (`https://en.wikipedia.org/wiki/Barrett_reduction`).
It allows you to compute $x \bmod P$ several times faster than usual, where P is constant but is not known at compile time. Link to KACTL implementation with additional info:
`https://github.com/kth-competitive-programming/kactl/blob/main/content/various/FastMod.h`.

Code:

```
using u32 = unsigned int;
using u64 = unsigned long long;
using u128 = __uint128_t;
struct Barrett {
    u64 b, m;

    Barrett() : b(), m() {}
    Barrett(u64 _b) : b(_b), m(-1ULL / _b) {}

    u32 reduce(u64 x) {
        u64 q = (u64)((u128(m) * x) >> 64), r = x - q * b;
        return r - b * (r >= b);
    }
} BA;
```

Usage example:

```
u32 mult(u32 x, u32 y) {
    return BA.reduce((u64)x * y);
}

int main() {
    int n, P;
    cin >> n >> P;
    BA = Barrett(P);
    cout << mult(123456, 7890123) << endl;
    return 0;
}
```

Don't forget to construct an instance of Barrett before using it.

We do not guarantee that this problem is solvable without Barrett reduction or other fast multiplication methods.

Input

The only line contains two integers N and P ($2 \le N \le 250$, $2 < P < 10^9$). It is guaranteed that P is prime and $P \bmod 2^{16} = 1$.

Output

Print $N - 1$ integers — answers to the problem for all n from 2 to N.

Recall that, if probability can be expressed as an irreducible fraction u/v, and v is coprime with P, the probability modulo P is $(u \cdot v^{-1}) \bmod P$, where v^{-1} is the multiplicative inverse of v modulo P.

Example

standard input	*standard output*
10 998244353	1 1 532396989 328786831 443364983 567813846 34567523 466373946 474334062

Problem G. MIT

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	998 244 353 bytes

This problem is dedicated to current ICPC champions. From one of 2020 ICPC Champions and, hopefully, 2023 ICPC Champions to 2021 and, hopefully, 2022 ICPC Champions with love.

You are given an edge-weighted tree T on n vertices. Define d_{uv} to be the sum of weights on the only simple path between u and v in T. Consider a full weighted graph G, where the weight of the edge (u, v) is d_{uv}.

For every k between 1 and $\lfloor \frac{n}{2} \rfloor$, calculate the maximum possible weight of a matching of size k in graph G. Recall that a matching of size k is a set of k edges such that no two edges in this set have a common vertex.

Input

The first line contains one integer n ($2 \le n \le 100\,000$) — the size of the tree.

The next $n - 1$ lines describe the edges of the tree. The i-th of them contains three integers u_i, v_i, w_i ($1 \le u_i, v_i \le n$, $1 \le w_i \le 10^8$) meaning that there is an edge (u_i, v_i) with weight w_i in T.

It is guaranteed that the given edges form a tree.

Output

Print $\lfloor \frac{n}{2} \rfloor$ integers — maximum weights of matchings of the corresponding sizes in G.

Example

standard input	*standard output*
7 1 3 99 2 3 82 3 4 4 4 5 43 5 6 5 4 7 3	181 280 287

Problem H. Exact Subsequences

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Consider all binary strings that have exactly n different non-empty subsequences (different by contents). Sort the strings in lexicographic order. Find the k-th such string in this order.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 100$) — the number of test cases. The descriptions of the t test cases follow.

The description of each test case consists of a single line with two integers n and k ($1 \le n, k \le 10^9$).

Output

For each test case, if there are less than k binary strings with exactly n different non-empty subsequences, print `-1` on a single line. Otherwise, print lexicographically k-th of them on the next two lines in the following format:

A non-empty binary string can be uniquely described by its first character and list of sizes of blocks of equal characters. You should print m and c on the first line, where m is the number of blocks and c is the first character. Then, on the second line, print the sizes of blocks $L_1, L_2, \ldots, L_m$ in order.

Example

standard input	*standard output*
8 3 1 3 2 3 3 3 4 3 5 1000000000 1 99824 4353 2129721 207087	1 0 3 2 0 1 1 2 1 1 1 1 1 3 -1 1 0 1000000000 11 0 9 2 2 1 6 2 1 2 7 1 1 9 0 9 9 8 2 4 4 3 5 3

Note

The actual strings corresponding to answers to the sample are:

```
000
01
10
111
-1
000...000 (1000000000 times)
00000000011001000000110110000000010
00000000011111111100000000110000111100011111000
```

Problem I. SPPPSPSS.

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

`SPPPSPSS.` stands for `Sort Permutation Performing Prefix Sort Plus Suffix Sort.`

You are given a permutation p of length n. You want to sort it in increasing order using the minimum number of operations. In the k-th operation you need to choose either the prefix of length k or the suffix of length k, and sort it in increasing order.

Input

The first line contains one integer n ($1 \le n \le 10^6$) — the size of the permutation.

The second line contains the permutation $p_1, p_2, \ldots, p_n$.

Output

Suppose the minimum number of operations needed to sort the given permutation is equal to m. Then you should print a string of length $m + 1$, the last character should be "`.`", and all other characters should be either "`P`" or "`S`" describing whether you want to sort prefix ("`P`") or suffix ("`S`") in the respective operation.

Examples

standard input	*standard output*
3 1 2 3	.
2 2 1	SP.
9 3 2 4 1 5 6 7 9 8	SSSP.
10 2 9 5 7 10 6 3 1 8 4	SPPPSPSS.

Note

This is how the permutation will change in the fourth sample:

Before	Operation	After
2 9 5 7 10 6 3 1 8 4	S : Sort suffix of length 1	2 9 5 7 10 6 3 1 8 4
2 9 5 7 10 6 3 1 8 4	P : Sort prefix of length 2	2 9 5 7 10 6 3 1 8 4
2 9 5 7 10 6 3 1 8 4	P : Sort prefix of length 3	2 5 9 7 10 6 3 1 8 4
2 5 9 7 10 6 3 1 8 4	P : Sort prefix of length 4	2 5 7 9 10 6 3 1 8 4
2 5 7 9 10 6 3 1 8 4	S : Sort suffix of length 5	2 5 7 9 10 1 3 4 6 8
2 5 7 9 10 1 3 4 6 8	P : Sort prefix of length 6	1 2 5 7 9 10 3 4 6 8
1 2 5 7 9 10 3 4 6 8	S : Sort suffix of length 7	1 2 5 3 4 6 7 8 9 10
1 2 5 3 4 6 7 8 9 10	S : Sort suffix of length 8	1 2 3 4 5 6 7 8 9 10

Problem J. Kth Lex Min Min Min Subpalindromes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Consider all arrays with length n consisting of integers from 1 to m. Let P be the minimum number of *continuous subarrays that are palindromic* one such array can have. Recall that an array is palindromic if it is equal to its own reverse.

Find the k-th lexicographically minimal array with P continuous subarrays that are palindromic. We are still only considering arrays with length n consisting of integers from 1 to m.

In other words, let's take all arrays with length n consisting of integers from 1 to m, leave only those of them that have the minimum number of continuous subarrays that are palindromic, and sort them lexicographically. Your task is to find k-th of them in this order.

Input

The only line of input contains three integers n, m and k ($1 \le n \le 10^6$, $1 \le m \le 10^6$, $1 \le k \le 10^{18}$).

Output

If there are less than k valid arrays, print `-1`. Otherwise, print the k-th lexicographically minimal of them.

Examples

standard input	*standard output*
1 1 1	1
2 2 2	2 1
3 3 3	2 1 3
9 9 8244353	2 4 1 2 6 8 1 2 7
10 7 998244353	-1
3 1000 994253860	998 244 353

Note

Did we put min number of min in the title? Min.

Problem K. 4

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

I could have asked you to calculate the number of anti-K_4 subgraphs, but that would be just solving this problem and copying problem K from GP of Nanjing 2021 (`https://codeforces.com/gym/103470/problem/K`) (solution from ecnerwala — `https://codeforces.com/blog/entry/97762?#comment-866645`), and why would I do this?

You are given a simple undirected graph. Calculate the number of its K_4 subgraphs (sets of 4 vertices such that there are all 6 edges between them in the graph).

Input

A simple graph. Come on. You got this. $4 \le n \le 100\,000$, $0 \le m \le 100\,000$. No self-loops or parallel edges, I promise.

Output

This problem uses a standard checker.

Examples

standard input	*standard output*
5 9 1 2 1 3 1 4 1 5 2 3 2 4 2 5 3 4 3 5	2
4 0	0

Problem L. 5

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	555 mebibytes

You are given an array a of length n consisting of non-negative integers. Calculate the number of pairs (k, T) such that there exists a subsequence of a of length k whose sum is equal to T.

Just kidding, this is too general. Suppose the sum of elements of a is equal to S, then it is guaranteed that a has at least $S/5$ elements equal to 1.

Input

The first line contains two positive integers n and S ($1 \le n, S \le 2 \cdot 10^5$) — the number of elements in a and their sum.

The second line contains the array $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le S$). It is guaranteed that $\sum_{i=1}^{n} a_i = S$ and at least $S/5$ elements of a are equal to 1.

Output

Print the number of pairs (k, T) such that there exists a subsequence of a of length k whose sum is equal to T.

Examples

standard input	*standard output*
7 9 0 0 0 1 1 2 5	42
10 33 9 9 8 1 1 1 1 1 1 1	48
10 14 2 4 4 1 0 1 0 1 0 1	81
10 14 3 5 3 0 1 0 1 0 1 0	87