Problem A. Binary Strings

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Given n non-empty binary strings $s_1, s_2, \ldots, s_n$ and another m non-empty binary strings $t_1, t_2, \ldots, t_m$, determine if there exists such a binary string S that:

- There exist i and j such that $1 \le i < j \le n$, and both strings s_i and s_j appear in S as substrings.
- For all i such that $1 \le i \le m$, string t_i does not appear in S as a substring.

Input

The first line contains one integer T ($1 \le T \le 10^5$) denoting the number of test cases. For each test case:

The first line contains two integers n and m ($2 \le n \le 10^5$, $1 \le m \le 10^5$).

The following n lines contain non-empty binary strings $s_1, s_2, \ldots, s_n$, one per line.

The following m lines contain non-empty binary strings $t_1, t_2, \ldots, t_m$, one per line.

For the total sums over all test cases, it is guaranteed $\sum n + \sum m \le 10^5$ and that $\sum |s_i| + \sum |t_i| \le 10^6$.

Output

For each test case, output a line containing a single string: "`Yes`" (without quotes) if such a binary string S exists, or "`No`" (without quotes) if not.

Example

standard input	*standard output*
2 3 2 100 001 010 1001 000 2 4 100 001 010 1001 000 11	Yes No

Note

For the first case, one possible string is "`0100`", where $s_1 =$ `100` and $s_3 =$ `010` appear in it, but $t_1 =$ `1001` and $t_2 =$ `000` don't appear.

Problem B. Collinear Arrangements

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

Given a convex polygon of n points $P_1, P_2, \ldots, P_n$ on a two-dimensional plane, answer q queries, where each query has one of the following types:

1. Given one point (x, y), find the number of pairs (P_i, P_j) such that $1 \le i < j \le n$ and the three points (x, y), P_i, and P_j are collinear.
2. Given two points (x_1, y_1) and (x_2, y_2), find the number of points P_i such that $1 \le i \le n$ and the three points (x_1, y_1), (x_2, y_2), and P_i are collinear.

Input

The first line contains two integers n and q ($3 \le n \le 10^5$, $1 \le q \le 10^5$) denoting the number of vertices in the given polygon and the number of queries, respectively.

Each of the following n lines contains two integers, x and y, denoting a vertex of the polygon.

Each of the following q lines contains one query, which is in one of the following formats:

1. "`1` x y", asking to calculate the number of pairs (P_i, P_j) such that $1 \le i < j \le n$ and the three points (x, y), P_i, and P_j are collinear.
2. "`2` x_1 y_1 x_2 y_2", asking to calculate the number of points P_i such that $1 \le i \le n$ and the three points (x_1, y_1), (x_2, y_2), and P_i are collinear.

It is guaranteed that:

- $|x|, |y| \le 10^9$ for all points and queries;
- the polygon vertices are given in counter-clockwise order;
- the polygon is convex (in particular, no three vertices are collinear);
- for each query, the given points and the polygon vertices do not coincide;
- the number of queries in the first format does not exceed 100.

Output

For each query, output a line containing a single integer: the answer to the query.

Example

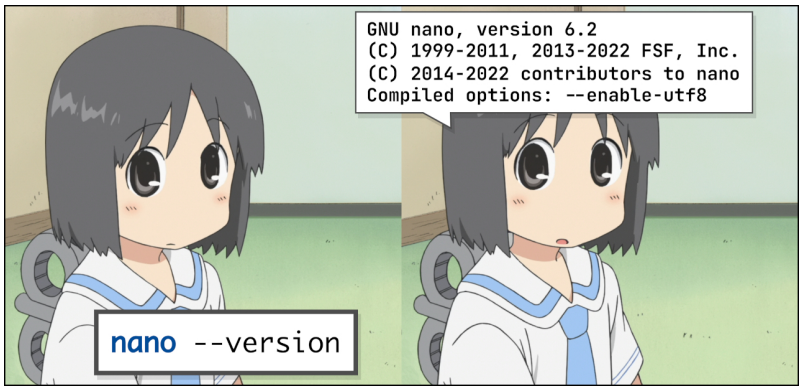
standard input	*standard output*
5 3 0 0 2 0 2 1 1 2 0 2 1 1 1 2 1 1 2 2 1 2 2	1 1 2

Note

- For the first query, the only pair is (P_2, P_5) since $(1, 1)$, $P_2 = (2, 0)$ and $P_5 = (0, 2)$ are collinear.
- For the second query, the only point is P_1 since $(1, 1)$, $(2, 2)$, and $P_1 = (0, 0)$ are collinear.
- For the third query, the two pairs are (P_2, P_3) and (P_4, P_5).

Problem C. Comedy's Not Omnipotent

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes



This is an interactive problem.

Vim, Emacs, and Nano are playing a guessing game. Vim secretly told Nano a **random** binary sequence $\{a_i\}$ of length n. Emacs can query Nano with a set of indices $I \subseteq \{1, 2, \ldots, n\}$. Nano will reply with $\sum_{i \in I} a_i$. Could you please help Emacs find $\{a_i\}$ in **less than** $n/2$ queries? Additionally, the total size of the sets in all queries **must not** be greater than $3n$.

Interaction Protocol

The first line of input contains an integer n.

You can use any of the following operations and write it to standard output:

1. "`?` k i_1 i_2 $\ldots$ i_k": Send a query with $I = \{i_1, i_2, \ldots, i_k\}$. The elements must be **distinct**. Nano will write the result back to standard input. There must be **less than** $n/2$ queries, and the sum of k for all queries **must not** be greater than $3n$.
2. "`=` $a_1 a_2 \ldots a_n$": Submit the binary sequence $\{a_i\}$ you found. Note that there are no spaces between a_i. Your program must exit gracefully after this operation.

Remember to **end the line** and **flush** the standard output after each operation. For example, you can use the function `fflush(stdout)` in C or C++, `System.out.flush()` in Java, `flush(output)` in Pascal, or `sys.stdout.flush()` in Python.

Example

standard input	*standard output*
4 2 1 2	? 4 1 2 3 4 ? 2 1 2 ? 2 2 3 = 0110

Note

The size $n = 10^5$ **in all tests**. The example with $n = 4$ shows the format but **will not be tested**.

There are at most 50 tests in this problem. The tests were generated randomly, but are fixed in advance. In each test, every binary sequence of length n had the probability of $1/2^n$ to be generated.

Problem D. Deep Abyss

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Kieray has a hash function $h(x)$. She wants you to help her find a fixed point x_0 of h, such that $h(x_0) = x_0$.

Input

The input contains the description of $h(x)$.

The function $h(x)$ is described by a procedure in Frog language. In this language, variables (for example, x, y, z) are 128-bit unsigned integers. At the beginning, the variable x contains the input value to h, and the other variables are all initialized to zeroes. The result of $h(x)$ is the value of variable x after the execution of the procedure.

The procedure contains at most 500 lines. Each line is an assignment that stores the result of an expression into a variable. Frog language supports bitwise negation (`~`), and (`&`), or (`|`), xor (`^`), and left/right shifting (`<<`/`>>`). The semantics of these operations are similar to those in C/C++.

Each assignment contains at most one binary operation. (Hexa)decimal constant numbers are allowed in expressions. The bitwise negation (`~`) can be applied to each variable or constant at most once. Every expression except bitwise xor expression (`^`) contains at most one variable. The shift amount (right operand) of a shifting operation is non-negative and not greater than 128.

Here is the extended BNF specification of the Frog language:

```
Newline      = "\n"
DecimalDigit = "0" ... "9"
HexDigit     = "0" ... "9" | "a" ... "f"
Letter       = "a" ... "z"
Hexadecimal  = "0" "x" HexDigit{1,32}
Variable     = Letter{1,4}
Term         = ( "~" | ) ( Variable | Hexadecimal )
ConstantTerm = ( "~" | ) Hexadecimal
ShiftAmount  = DecimalDigit
             | ( "1" ... "9" ) DecimalDigit
             | "1" ( "0" | "1" ) DecimalDigit
             | "1" "2" ( "0" ... "8" )
Expression   = Term
             | Term "␣" "^" "␣" Term
             | Term "␣" "&" "␣" ConstantTerm
             | ConstantTerm "␣" "&" "␣" Term
             | Term "␣" "|" "␣" ConstantTerm
             | ConstantTerm "␣" "|" "␣" Term
             | Term "␣" "<<" "␣" ShiftAmount
             | Term "␣" ">>" "␣" ShiftAmount
Assignment   = Variable "␣" "=" "␣" Expression
Procedure    = ( Assignment Newline )*
```

Note: "`*`" means the preceding token appears zero or more times, and "`{1,4}`" means the preceding token appears one to four times. The symbol "␣" represents a single space character.

Output

Print the fixed point of $h(x)$ in hexadecimal without leading zeroes, conforming to the format of `Hexadecimal` in the extended BNF specification. If there are multiple fixed points, print the minimum one. If there is no fixed point, print ":(" (without quotes) instead.

Examples

standard input	*standard output*
`x = x | 0x19260817`	`0x19260817`
`x = x ^ 0xdeadbeef`	`:(`

`y = ~x << 3`
`z = ~x >> 2`
`w = x & ~0xa`
`k = ~x ^ 0xc`
`k = k << 1`
`k = k >> 2`
`x = y ^ z`
`p = k ^ w`
`x = x ^ p`
	`0x9a83dcd41ee6a0f73507b9a83dcd41ef`

`x = x | 0x1` `x = x << 128`	`0x0`

Note

This is an illustration from Kieray. It's lovely.

Problem E. Lines on a Phone Screen

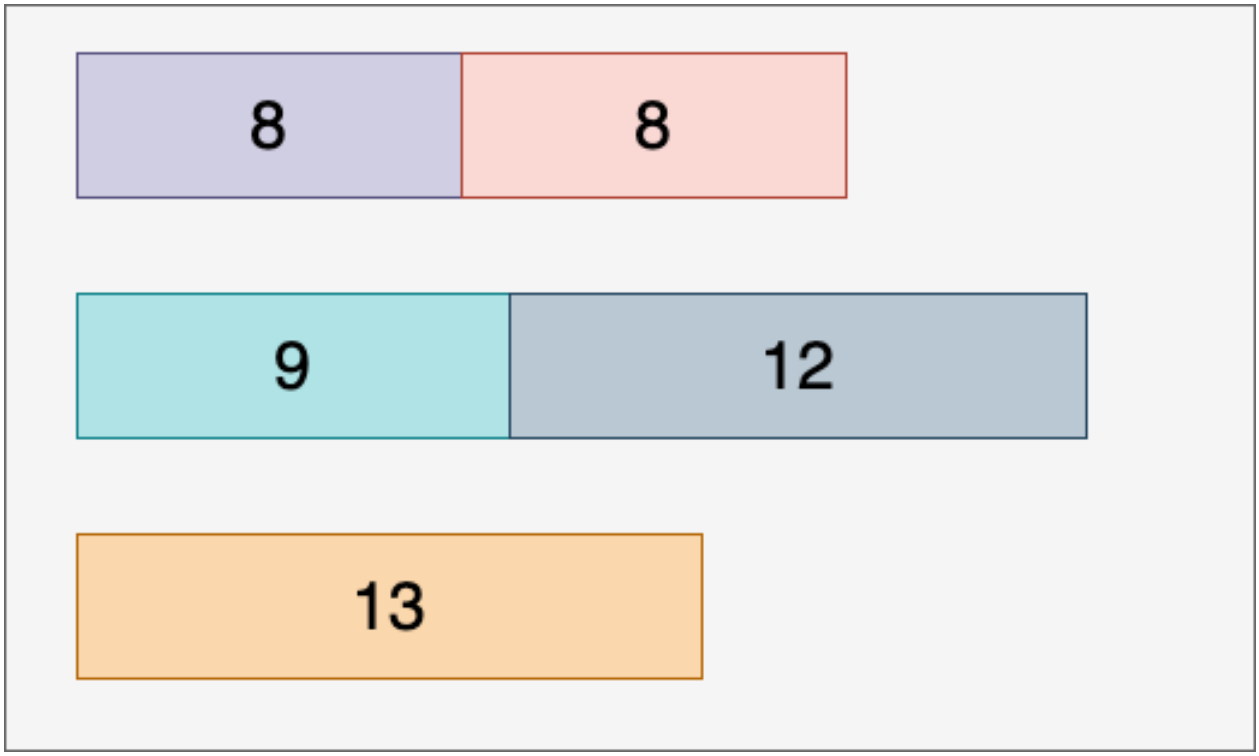
Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Suppose that there is a sequence of k sentences in a mobile phone. To describe the length of each sentence, we use an integer sequence $S = (s_1, s_2, \ldots, s_k)$, where s_i is the length of the i-th sentence. It is guaranteed that $1 \le s_i \le 24$.

To show users the sentences, the mobile phone will print them in order. However, because of the width limitation of the screen, the total length of the sentences in one line should not exceed 24. Moreover, to make sentences reader-friendly, every sentence should be put in exactly one line (in other words, a sentence cannot be split into multiple lines).

The rule to satisfy the requirements is as follows. The first sentence is printed on the first line. For $i \ge 2$, the mobile phone prints the i-th sentence in the last line if the line length does not exceed 24 after that. Otherwise, it starts a new line and prints the i-th sentence in it.

For example, if $S = (8, 8, 9, 12, 13)$, the sentences will be printed like this:



Now, here is the problem.

There are n sentences, and their lengths are $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 24)$.

And there are m operations to do. Each operation belongs to one of the two types below:

- *op1*(x, c): Change the length of the x-th sentence from a_x to c.
- *op2*(ℓ, r): Determine the number of lines on the screen when printing the sentences with lengths $S = (a_\ell, a_{\ell+1}, a_{\ell+2}, \ldots, a_r)$ on the mobile phone.

Your task is to answer all questions given by the operations of the second type.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^5$) denoting the number of sentences and the number of operations, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 24$) denoting the lengths of the sentences.

Then m lines follow. Each of them contains three integers and represents one of the two types of operations:

- "`1` x c": Change the length of the x-th sentence from a_x to c ($1 \le x \le n$, $1 \le c \le 24$).
- "`2` ℓ r": Print the number of lines when printing sentences from ℓ-th to r-th on the screen ($1 \le \ell \le r \le n$).

Output

For each operation of the second type, print a single line with the answer.

Example

standard input	*standard output*
5 5 8 8 9 12 13 2 1 5 2 2 4 1 5 3 2 1 5 2 2 5	3 2 2 2

Problem F. Majority

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

Little Cat learned Boolean circuits recently. Now he wants to construct a majority circuit.

A *circuit* over Boolean variables $x_1, \ldots, x_n$ is a directed acyclic graph where each node (logical gate) is either an input node labeled by a variable x_i, or an operation node labeled by a logical operation $\vee$ or $\wedge$. There are exactly n input nodes, one for each of the input variables $x_1, \ldots, x_n$. Additionally, a single node is **chosen** as the output of the circuit.

Each node computes an output: the input nodes (labeled by a variable) output exactly the variable written on them, and nodes labeled by $\vee$ (respectively, $\wedge$) output the logical OR (respectively, AND) of all incoming nodes. Note that logical NOT nodes are **forbidden**. See the example and notes for better understanding.

The in-degree of an input node is 0. The in-degree of an operation node is at least 1, and can be arbitrarily large. The out-degrees are arbitrary (possibly 0).

For convenience, there are two special constant nodes T (true) and F (false), which always output 1 and 0, respectively.

The majority circuit Maj_n has n inputs $x_1, \ldots, x_n$, and it outputs 1 if at least half of inputs are 1, and outputs 0 otherwise. Formally, $Maj_n(x_1, \ldots, x_n) = [2\sum_{i=1}^{n} x_i \geq n]$.

Define the *depth* of a circuit as the length of the longest (directed) path in the circuit, that is, the number of edges of the longest path.

Could you help Little Cat to construct a majority circuit over n inputs with depth at most 14?

Input

The input contains one line with an integer n ($2 \leq n \leq 64$) indicating the number of input nodes.

Output

The first line must contain an integer m ($1 \leq m \leq 5 \cdot 10^4$) representing the number of nodes labeled by $\vee$ or $\wedge$, so there are $n + m + 2$ nodes in the circuit in total. The input nodes $x_1, \ldots, x_n$ are numbered by $1, \ldots, n$. The constant true node T is numbered by -1, and the constant false node F is numbered by -2.

A total of m lines must follow. The i-th line must describe node $(n + i)$ in one of the following formats.

- "`OR` k_i a_1 a_2 ... a_{k_i}" (without quotes): node $(n + i)$ computes the logical OR of nodes a_j where $-2 \leq a_j < n + i$ and $a_j \neq 0$ for all $1 \leq j \leq k_i$.
- "`AND` k_i a_1 a_2 ... a_{k_i}" (without quotes): node $(n+i)$ computes the logical AND of nodes a_j where $-2 \leq a_j < n + i$ and $a_j \neq 0$ for all $1 \leq j \leq k_i$.

It is fine if $a_u = a_v$ for some $u \neq v$. You must guarantee that $\sum_{i=1}^{m} k_i \leq 2 \cdot 10^5$ and that the depth of the circuit does not exceed 14.

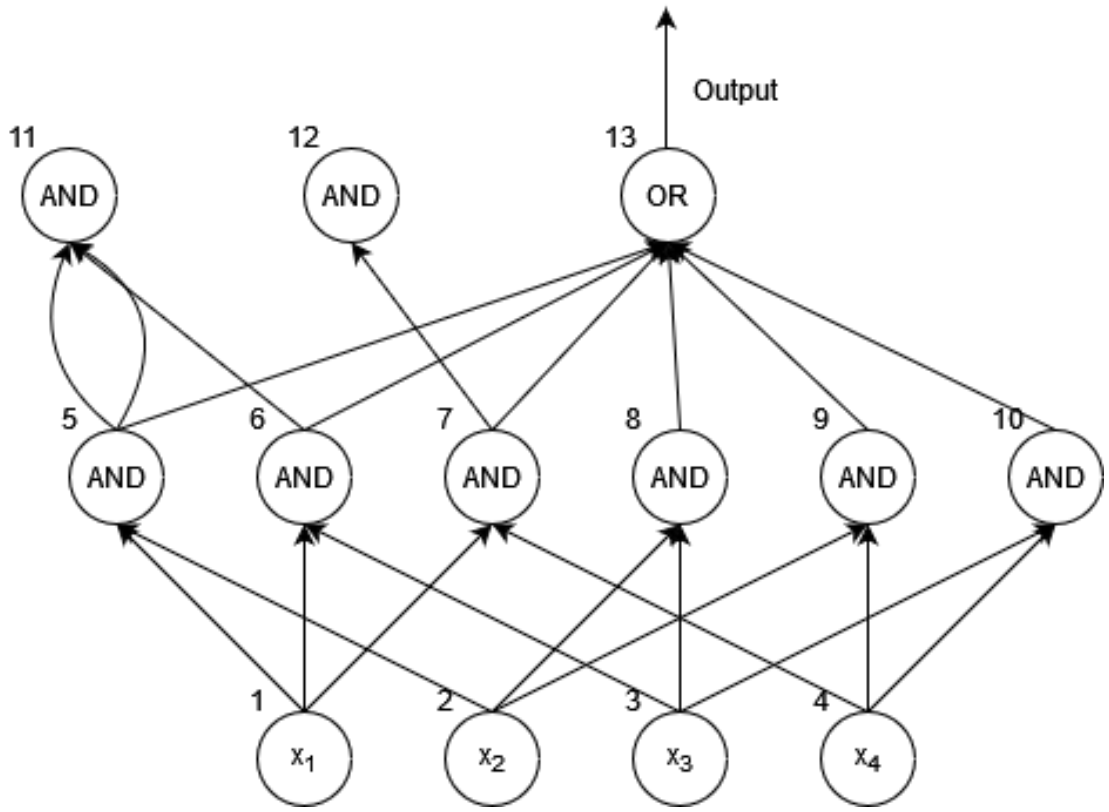
The output of the circuit **is chosen as** the output of node $n + m$.

To check the circuit you construct, Little Cat will test your circuit for 1500 rounds. In each round, Little Cat will generate an arbitrary input $x_1, \ldots, x_n$ (he won't say how exactly) and test your circuit with that input. You pass this round if your circuit outputs the majority of the input $x_1, \ldots, x_n$ correctly. You need to pass all the 1500 rounds.

Example

standard input	*standard output*
4	9 AND 2 1 2 AND 2 1 3 AND 2 1 4 AND 2 2 3 AND 2 2 4 AND 2 3 4 AND 3 5 5 6 AND 1 7 OR 6 5 6 7 8 9 10

Note



The sample output prints a depth-2 circuit computing Maj_4. The circuit $Maj_4(x_1, x_2, x_3, x_4)$ outputs 1 if and only if at least two input nodes are 1. Thus you can compute the logical AND of every pair of input nodes and output the logical OR of these ANDs.

Here are some notes on the circuit nodes:

- Nodes 1, 2, 3, and 4 are input nodes.
- Nodes 5, 6, 7, 8, 9, and 10 compute the logical AND of some input nodes.
- Nodes 11 and 12 are redundant.
- Node 13 is the output node.
- The constant nodes T and F are not drawn in the figure.

Here, the existence of redundant nodes will not affect the validity of the circuit as long as the constraints ($1 \le m \le 5 \cdot 10^4$, $\sum k_i \le 2 \cdot 10^5$, $depth \le 14$) are satisfied.

During the test, the following shows a possible scenario:

The input nodes are set to $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, $x_4 = 1$.

Therefore, the outputs of nodes $x_5, \ldots, x_{13}$ are:

- $x_5 = x_1$ AND $x_2 = 1$ AND $0 = 0$
- $x_6 = x_1$ AND $x_3 = 1$ AND $0 = 0$
- $x_7 = x_1$ AND $x_4 = 1$ AND $1 = 1$
- $x_8 = x_2$ AND $x_3 = 0$ AND $0 = 0$
- $x_9 = x_2$ AND $x_4 = 0$ AND $1 = 0$
- $x_{10} = x_3$ AND $x_4 = 0$ AND $1 = 0$
- $x_{11} = x_5$ AND x_5 AND $x_6 = 0$ AND 0 AND $0 = 0$
- $x_{12} = x_7 = 1$
- $x_{13} = x_5$ OR x_6 OR x_7 OR x_8 OR x_9 OR $x_{10} = 0$ OR 0 OR 1 OR 0 OR 0 OR $0 = 1$.

The output of the circuit is $x_{13} = 1$, which is the majority of $\{1, 0, 0, 1\}$.

Problem G. Matrices and Determinants

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Given an $n \times n$ integer matrix A, you should find two $n \times n$ integer matrices B and C such that $B \cdot C = A$ and $\det(B) = \det(C) \neq 0$. There may exist multiple solutions or no solution.

Note: $\det(M)$ denotes the determinant of matrix M.

Input

The first line contains one integer T ($1 \leq T \leq 10\,000$) denoting the number of test cases. For each test case:

The first line contains one integer n ($1 \leq n \leq 4$) denoting the size of the given matrix.

In the following n lines, the i-th line contains n integers $A_{i,j}$ ($|A_{i,j}| \leq 10$ for $1 \leq j \leq n$) denoting the given matrix.

Output

For each test case:

The first line must contain one string "`Yes`" (without quotes) if a solution exists, or "`No`" (without quotes) if there is no solution. If a solution exists:

Each of the following n lines contains n integers $B_{i,j}$ ($|B_{i,j}| \leq 10^{18}$) denoting the matrix B.

Each of the following n lines contains n integers $C_{i,j}$ ($|C_{i,j}| \leq 10^{18}$) denoting the matrix C.

If multiple solutions exist, print any one of them.

Example

standard input	*standard output*
3 2 2 0 0 2 2 2 1 1 2 1 1	Yes 2 0 0 1 1 0 0 2 No Yes -1 -1

Problem H. Matrices and Sums

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Given a positive integer n, you should construct an $n \times n$ integer matrix M satisfying the following conditions:

- For all elements $M_{i,j}$ $(1 \le i, j \le n)$, the absolute value $|M_{i,j}| \le 1$.
- The row and column sums $R_1, R_2, \ldots, R_n, C_1, C_2, \ldots, C_n$ are pairwise distinct, where $R_x = \sum_{i=1}^{n} M_{x,i}$ and $C_x = \sum_{i=1}^{n} M_{i,x}$.

There may exist multiple solutions or no solution.

Input

The first line contains a single integer n $(1 \le n \le 1000)$.

Output

The first line must contain one string "`Yes`" (without quotes) if a solution exists, or "`No`" (without quotes) if there is no solution.

When a solution exists, print n more lines, each containing n integers, denoting the matrix M you construct.

If multiple solutions exist, print any one of them.

Examples

standard input	*standard output*
2	Yes 1 0 1 -1
1	No

Note

- In the first example, $R_1 = 1$, $R_2 = 0$, $C_1 = 2$, and $C_2 = -1$ are all distinct.
- In the second example, $R_1 = C_1$ always holds, so no solution exists.

Problem I. Palindrome Strings

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

You are given a string $S = S_1S_2 \dots S_{|S|}$ and q queries. In each query, a string $t = t_1t_2 \dots t_{|t|}$ is given, and you should determine the number of pairs (ℓ, r) such that $1 \le \ell \le r \le |S|$ and the combined string $t_1t_2 \dots t_{|t|}S_\ell S_{\ell+1} \dots S_r$ is a palindrome, which means that

$$t_1t_2 \dots t_{|t|}S_\ell S_{\ell+1} \dots S_r = S_rS_{r-1} \dots S_\ell t_{|t|}t_{|t|-1} \dots t_1 .$$

Input

The first line contains two integers n and q ($1 \le n \le 10^6$, $1 \le q \le 10^5$) denoting the length of string S and the number of queries, respectively.

The second line contains a single string S.

Each of the following q lines contains a single string t denoting a query.

It is guaranteed that all the strings only contain lowercase English letters and that $\sum |t| \le 10^6$.

Output

For each query, output a single line containing one integer: the required number of pairs.

Example

standard input	*standard output*
8 3 icpccamp p c pc	4 7 4

Note

- For the first query, the 4 pairs are $(2, 3)$, $(3, 3)$, $(7, 8)$, and $(8, 8)$, and the combined strings are "pcp", "pp", "pmp", "pp", respectively.
- For the second query, the 7 pairs are $(1, 2)$, $(2, 2)$, $(2, 5)$, $(3, 4)$, $(4, 4)$, $(4, 5)$, and $(5, 5)$.
- For the third query, the 4 pairs are $(1, 3)$, $(2, 3)$, $(3, 3)$, and $(8, 8)$.

Problem J. Apple Family Reunion

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Background

One rainy night, a hooded mare gallops into an empty marketplace and enters a darkened curio shop. She begins to rummage around the shop until a small candle is lit.

"May I help you, traveler?" A stallion, who is the shopkeeper, comes out and asks. Stepping behind the counter, he surmises that something powerful drew the customer to his shop; she points to an amulet inlaid with a crimson gem kept under glass on the shelf behind him.

"Ah, you have a keen eye. The Alicorn Amulet is one of the most mysterious and powerful of all the known magical charms. Uh, ah — I'm afraid this is... far too dangerous."

(The mare tosses a big bag of gold bits onto the counter.)

"Would you like that gift-wrapped?"

...

The next morning, all ponies awake to find themselves changed into permutations!

There are $n!$ ponies in Equestria. The evil magic turned all ponies into $n!$ different permutations of length n.

In order to reduce the panic, Applejack, who changed into permutation A, decides to gather her family as soon as possible. She notices that, for another permutation P, if she can change herself into P with several applications of One-Two-Three-transformation, then P is a member of the Apple family.

The One-Two-Three-transformation goes as follows. Choose 3 adjacent elements A_i, A_{i+1}, A_{i+2}. If A_i is the median of them, you can put it behind A_{i+2}, in other words, transform these three elements into A_{i+1}, A_{i+2}, A_i. If A_{i+2} is the median of them, you can put it in front of A_i, in other words, transform these three elements into A_{i+2}, A_i, A_{i+1}.

In fact, if a permutation could change into another with One-Two-Three-transformations, then they belong to the same family. Let us choose the lexicographically smallest permutation in each family as the representative of that family. Then we can sort all families by the lexicographical order of their representatives, obtaining the list of families $F_1, F_2, \ldots$. For example, the permutation $(1, 2, \ldots, n)$ always belongs to the family F_1.

Each pony should go to the temporary shelter of their family. A permutation $P \in F_i$ implies that pony P should go to shelter i to meet its family.

Please help Applejack find which shelter the Apple family should go to, and how many ponies are in the Apple family.

Input

The first line contains one integer n ($3 \le n \le 10^5$).

The second line contains n integers $A_1, A_2, \ldots, A_n$, denoting the permutation A Applejack changed into.

Output

Output one line containing one integer.

We know there exists a unique i such that $A \in F_i$. **If $i \le 65\,472$, print the number of ponies in the Apple family modulo $998\,244\,353$. Otherwise, just print i modulo $998\,244\,353$ as the answer.**

Example

standard input	*standard output*
3 2 1 3	2

Note

There are 4 families in total:

- $F_1 = \{(1, 2, 3)\}$
- $F_2 = \{(1, 3, 2),\ (2, 1, 3)\}$
- $F_3 = \{(2, 3, 1),\ (3, 1, 2)\}$
- $F_4 = \{(3, 2, 1)\}$

So the Apple family is $F_2 = \{(1, 3, 2),\ (2, 1, 3)\}$, and their representative is $(1, 3, 2)$. So, print the number of ponies in the Apple family, which is 2.

Problem K. Twinning Totem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

"Ori, this is a Map Stone, one of the many ancient markers created to chart the forest of Nibel as it grew."

The Map Stone that Ori meets today is different from the others. The totem has two sides, and each side has n hollows with m scratches. Each scratch connects two hollows, and the hollows and the scratches on the two sides are the same. Each hollow can pass through the stone so that, if we put something on one side, we could also see it on the other side. However, the scratches are the same but cannot pass through, so we can put different things on two sides for each scratch.

Ori puts one Life Cell into hollow u and $n-1$ Energy Cells into the other $n-1$ hollows. Now, on each side, Ori needs to select $n-1$ scratches and inject light to them to form a spanning tree rooted in u. The two resulting trees should have the following property: for each Energy Cell, if it flows its energy to the Life Cell along light scratches on one side, and then flows back along the light scratches on the other side, no other Energy Cell will be passed twice.

Could Ori find two trees with the required properties?

Input

The first line contains two integers n and m ($3 \le n \le 10^5$, $n-1 \le m \le \min(2 \cdot 10^5, \frac{n(n-1)}{2})$) denoting the number of hollows and the number of scratches of the totem G, respectively.

For the next m lines, the i-th line contains two integers x_i and y_i ($1 \le x_i, y_i \le n$) denoting a scratch connecting hollow x_i and hollow y_i.

The next line contains an integer T ($1 \le T \le 10^5$) denoting the number of Ori's attempts.

For the next T lines, the i-th line contains an integer u_i ($1 \le u_i \le n$) denoting the hollow where Ori puts the Life Cell in the i-th attempt.

It is guaranteed that G is a connected graph without self-loops or multiple edges, and that $n \cdot T \le 10^5$.

Output

For each attempt of Ori, if no solution exists, output "`No`" (without quotes) on the first line.

Otherwise, output "`Yes`" (without quotes) on the first line. Then, output $2(n-1)$ lines to describe the two spanning trees.

For the first $n-1$ of these lines, the i-th line must contain two integers x_i and y_i ($1 \le x_i, y_i \le n$) separated by a space, denoting a selected scratch connecting x_i and y_i. The $n-1$ scratches must form the first spanning tree of G.

The next $n-1$ lines must describe the second spanning tree of G in the same format.

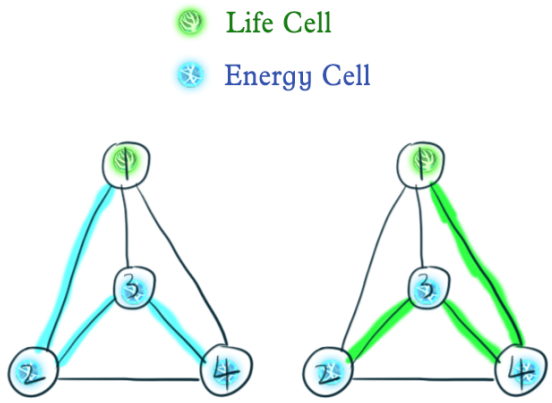
Your answer will be accepted only when the two trees you output are valid spanning trees of G and meet the requirements in the problem description: for any $v \in V$ such that $v \neq u$, the two simple paths from u to v in the two trees have no common nodes except their endpoints u and v.

Examples

standard input	*standard output*
4 6 1 2 1 3 1 4 2 3 2 4 3 4 1 1	Yes 1 2 2 3 3 4 3 2 4 3 1 4
4 4 1 3 2 3 2 4 3 4 4 1 2 3 4	No No Yes 3 1 4 2 3 4 3 1 3 2 2 4 No

Note

For the first example:



- For hollow 2, the paths in the two trees are $P_1 = \{2, 1\}$ and $P_2 = \{2, 3, 4, 1\}$.
- For hollow 3, the paths in the two trees are $P_1 = \{3, 2, 1\}$ and $P_2 = \{3, 4, 1\}$.
- For hollow 4, the paths in the two trees are $P_1 = \{4, 3, 2, 1\}$ and $P_2 = \{4, 1\}$.

Problem L. Count the Orders

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

There are n positions on a circle, numbered successively by integers from 1 to n. The positions i and $i+1$ are adjacent; the positions n and 1 are also adjacent.

Consider n distinct integers $a_1, a_2, \ldots, a_n$. We arrange them somehow on the circle, so that there is a single integer in each of the n positions. The cost of an arrangement is defined as the sum of the absolute values of the difference between every two adjacent integers.

Two arrangements are different if and only if at least one integer has different positions in them.

You need to find the maximum cost of an arrangement. Additionally, calculate the number of different arrangements that have this cost. As their number can be very large, find it modulo $10^9 + 7$.

Input

The first line contains a single integer n ($3 \le n \le 10^6$).

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

It is guaranteed that $a_1, a_2, \ldots, a_n$ are pairwise distinct.

Output

Output a single line with two integers. The first one should be the maximum cost. The second one should be the number of different arrangements that have this cost, modulo $10^9 + 7$.

Examples

standard input	*standard output*
3 1 2 3	4 6
4 2 4 8 16	36 8

Problem M. Simple Math Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Given two positive integers m and n, determine the value of the following formula modulo 998 244 353:

$$\sum_{i=0}^{\lfloor \frac{m}{2} \rfloor} \sum_{j=0}^{\lfloor \frac{n}{2} \rfloor} \binom{i+j}{j}^2 \binom{m+n-2i-2j}{n-2j}.$$

Here, $\binom{a}{b}$ is a binomial coefficient (the number of ways to choose an unordered subset of b items from a fixed set of a items).

Input

The first line contains one integer T ($1 \le T \le 10^5$) denoting the number of test cases.

For each test case, the input is a single line containing two integers m and n ($1 \le m, n \le 10^5$).

Output

For each test case, output one line containing one integer: the value of the formula modulo 998 244 353.

Example

standard input	*standard output*
2 1 9 2 6	30 80