Zhejiang U Summer Training Camp
Legilimens Contest

Chen, Ye, Liu

March 16, 2018

Problem A. Nearly Lucky Number

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2000 ms
Memory limit:	256 MB

Petya loves lucky numbers. We all know that lucky numbers are the positive integers whose decimal representations contain only the lucky digits 4 and 7. For example, numbers 47, 744, 4 are lucky and 5, 17, 467 are not.

Unfortunately, not all numbers are lucky. Petya calls a number nearly lucky if the number of lucky digits in it is a lucky number. He wonders whether number n is a nearly lucky number.

Input

The only line contains an integer $n(1 \leq n \leq 10^{18})$.

Output

Print on the single line "YES" if n is a nearly lucky number. Otherwise, print "NO" (without the quotes).

Examples

standard input	standard output
40047	NO
7747774	YES
1000000000000000000	NO

Problem B. Foe Pairs

Input file:	standard input
Output file:	standard output
Time limit:	1000 ms
Memory limit:	256 MB

You are given a permutation p of length n.

Also you are given m foe pairs (a_i, b_i) $(1 \le a_i, b_i \le n, a_i \ne b_i)$.

Your task is to count the number of different intervals (x, y) $(1 \le x \le y \le n)$ that do not contain any foe pairs. So you shouldn't count intervals (x, y) that contain at least one foe pair in it (the positions and order of the values from the foe pair are not important).

Consider some example: p = [1, 3, 2, 4] and foe pairs are $\{(3, 2), (4, 2)\}$. The interval $(1, 3)$ is incorrect because it contains a foe pair $(3, 2)$. The interval $(1, 4)$ is also incorrect because it contains two foe pairs $(3, 2)$ and $(4, 2)$. But the interval $(1, 2)$ is correct because it doesn't contain any foe pair.

Input

The first line contains two integers n and m $(1 \le n, m \le 3 \cdot 10^5)$ —the length of the permutation p and the number of foe pairs.

The second line contains n distinct integers p_i $(1 \le p_i \le n)$ —the elements of the permutation p.

Each of the next m lines contains two integers (a_i, b_i) $(1 \le a_i, b_i \le n, a_i \ne b_i)$ —the i_{th} foe pair. Note a foe pair can appear multiple times in the given list.

Output

Print the only integer c —the number of different intervals (x, y) that does not contain any foe pairs.

Note that the answer can be too large, so you should use 64-bit integer type to store it. In C++ you can use the long long integer type and in Java you can use long integer type.

Examples

standard input	standard output
4 2 1 3 2 4 3 2 2 4	5
9 5 9 7 2 3 1 4 6 5 8 1 6 4 5 2 7 7 2 2 7	20

Hint

In the first example the intervals from the answer are $(1, 1)$, $(1, 2)$, $(2, 2)$, $(3, 3)$ and $(4, 4)$.

Problem C. Pluses everywhere

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3000 ms
Memory limit:	256 MB

Vasya is sitting on an extremely boring math class. To have fun, he took a piece of paper and wrote out n numbers on a single line. After that, Vasya began to write out different ways to put pluses ("+") in the line between certain digits in the line so that the result was a correct arithmetic expression; formally, no two pluses in such a partition can stand together (between any two adjacent pluses there must be at least one digit), and no plus can stand at the beginning or the end of a line. For example, in the string 100500, ways 100500 (add no pluses), $1 + 00 + 500$ or $10050 + 0$ are correct, and ways $100 + +500$, $+1 + 0 + 0 + 5 + 0 + 0$ or 100500+ are incorrect.

The lesson was long, and Vasya has written all the correct ways to place exactly k pluses in a string of digits. At this point, he got caught having fun by a teacher and he was given the task to calculate the sum of all the resulting arithmetic expressions by the end of the lesson (when calculating the value of an expression the leading zeros should be ignored). As the answer can be large, Vasya is allowed to get only its remainder modulo $10^9 + 7$. Help him!

Input

The first line contains two integers, n and $k (0 \le k < n \le 10^5)$.

The second line contains a string consisting of n digits.

Output

Print the answer to the problem modulo $10^9 + 7$.

Examples

standard input	standard output
3 1 108	27
3 2 108	9

Problem D. Youngling Tournament

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2000 ms
Memory limit:	256 MB

Yoda, the Grand Master of the Jedi Order, hit on the idea to hold a tournament among younglings. He has not chosen the date yet, but he has already decided on the format of the tournament.

There are N younglings studying in the Jedi Temple. Yoda can feel the Force inside younglings, moreover, he can represent the amount of the Force inside i_{th} youngling as the number f_i.

Therefore, the format of the tournament is the following. At first, Yoda makes the children stand in a row in order of non-increasing f_i. Then, the first youngling in the row competes against all the others united together. The combat is based on lightsaber battle and is very spectacular. However, Yoda knows that the result doesn't depend on anything but the Force inside competitors. The youngling wins if his amount of Force is not less than the total amount of the Force inside all his opponents. In that case, he is considered one of the winners. Otherwise, he loses. Anyway, after that he is removed from the row, and the tournament continues. Again, the strongest (first in the row) youngling competes against all the others standing in the row in the same format, if he wins, he is also considered one of the winners. After that he is removed and the tournament continues in the same format until there is only one child in the row. He becomes one of the winners automatically and the tournament finishes.

Yoda wants to know the total number of winners. However, as the tournament is postponed again and again, the amount of the Force inside the younglings changes from time to time. Help Yoda to compute the total number of winners after each change.

Input

The first line of input contains a single integer N, the number of younglings in the Jedi Temple ($1 \le N \le 100\,000$).

The second line contains N integers $f_1, f_2, ..., f_N$ the amount of the Force inside the younglings ($1 \le f_i \le 10^{12}$).

The third line of the input contains a single integer M, the number of changes in the Force amounts of students ($0 \le M \le 50\,000$).

The next M lines contain information about the changes. The i_{th} of these lines describes the i_{th} change and contains two integers k and f_k^*, which mean that the amount of the Force inside the k_{th} youngling becomes equal to f_k^* ($1 \le k \le N, 1 \le f_k^* \le 10^{12}$).

Output

Print $M + 1$ lines, each of them containing a single integer.

On the first line, print the number of winners if the tournament was held before all the changes. On line $(i + 1)$ for all $i > 0$, print the number of winners if the tournament was held after the first i changes.

Examples

standard input	standard output
3 2 1 3 3 1 3 2 7 3 5	3 2 3 2
7 2 14 14 15 5 2 5 5 5 2 4 12 5 4 3 10 7 9	4 3 3 3 3 4

Problem E. Unique Party

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5000 ms
Memory limit:	256 MB

In a multi-storied building, there are some families who know each other very well. They like to have a lot of fun, so they often want to get together in one of those apartments to have a party. To understand the uniqueness of this group, let us consider a co-ordinate system where (x, y, z) denotes the co-ordinate of an apartment. Here, z axis denotes the oor of the apartment and (x, y) co-ordinates form the plane parallel to the land.

Now there is an interesting fact: there are no two families in their group such that their apartments have the same (x,y) co-ordinate. Moreover, for every possible value of (x,y), the there exists exactly one apartment in which one of these families live. Therefore, we can represent the oor of each of their apartments in a table as the following gure:

Here, you can consider the rows of the table parallel to x-axis and columns of the tables parallel to y-axis of the building. So, if the cell of 1st row and 1st column denotes 6, that means one of their friends live in the apartment on 6_{th} oor of the building with co-ordinate (1, 1).

It is not always possible for each family to be present in every party, but whenever a subset of these families gets together for a party, they have to satisfy the following properties:

- This subset is formed by a rectangle (axis-parallel) drawn on the table.
- The party will be held in one of the apartments of this subset (Obviously).
- The summation of oors they need to go up or down must be minimized. If there are multiple oors which satisfy this requirement, they will always choose the higher oor.
- After selecting the apartment, if it is on a oor lower than h, the party will be cancelled.

6	10	3	1
5	4	2	5
1	7	4	15

For example, say the families marked in grey (in the table shown above) want to hold a party and $h = 3$. Both 4_{th} and 5_{th} floor has the same summation of floors they need to go up or down($2+1+11 = 14$ for 4_{th} floor and $3 + 1 + 10 = 14$ for 5_{th} floor). As they always prefer the higher floor, it will be held on the 5_{th} floor. As it is not lower than 3_{rd} floor ($h = 3$), it also satisfies the last requirement.

You need to write a program to determine the the largest number of families who can get together for a party satisfying all those requirements.

Input

The first line will contain the number of test cases, T ($1 \le T \le 20$). Each test case starts with a line containing two integers, R ($1 \le R \le 250$) and C ($1 \le C \le 250$) denoting the number of rows and columns in the table, respectively. Then R lines follow, each containing C integers denoting the floors of corresponding apartments. Then there will be a line containing a single integer, Q ($1 \le Q \le 10$) which denotes the number of queries to follow. Then the following line contains Q integers, where each one denotes the value of h. All integers in the input file will be positive and in the range of 32-bit signed integer.

Output

For each case, print a line containing "Case x:" where x is the case number. Then the following Q lines should contain one integer: the largest number of families who can get together for the corresponding value of h.

Examples

standard input	standard output
1 3 4 6 10 3 1 5 4 2 5 1 7 4 15 2 6 5	Case 1: 6 12

Problem F. Almost Union-Find

Input file:	standard input
Output file:	standard output
Time limit:	1000 ms
Memory limit:	256 MB

I hope you know the beautiful Union-Find structure. In this problem, you're to implement something similar, but not identical. The data structure you need to write is also a collection of disjoint sets, supporting 3 operations:

1 p q	Union the sets containing p and q. If p and q are already in the same set, ignore this command.
2 p q	Move p to the set containing q. If p and q are already in the same set, ignore this command.
3 p	Return the number of elements and the sum of elements in the set containing p.

Initially, the collection contains n sets: $\{1\}, \{2\}, \{3\}, ..., \{n\}$.

Input

There are several test cases. Each test case begins with a line containing two integers n and m ($1 \leq n, m \leq 100000$), the number of integers, and the number of commands. Each of the next m lines contains a command. For every operation, $1 \leq p, q \leq n$. The input is terminated by end-of-file (EOF).

Output

For each type-3 command, output 2 integers: the number of elements and the sum of elements.

Examples

standard input	standard output
5 7 1 1 2 2 3 4 1 3 5 3 4 2 4 1 3 4 3 3	3 12 3 7 2 8

Hint

Initially: {1}, {2}, {3}, {4}, {5}

Collection after operation 1 1 2: {1,2}, {3}, {4}, {5}

Collection after operation 2 3 4: {1,2}, {3,4}, {5} (we omit the empty set that is produced when taking out 3 from {3})

Collection after operation 1 3 5: {1,2}, {3,4,5}

Collection after operation 2 4 1: {1,2,4}, {3,5}

Problem G. Handling a Spaceship

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1000 ms
Memory limit:	256 MB

This is an interactive problem.

Wedge Antilles has infiltrated a new secret imperial spaceship which uses N-dimensional hyperspace to move around the Galaxy faster than any other spaceship. Fortunately, there is nobody onboard now, but the spaceship is heading fast in some direction!

The spaceship uses N selectors to choose speed, each of them can be set in one of the M positions. The i_{th} selector corresponds to i_{th} direction X_i, where X_i is a N-dimensional vector; when it is set to j_{th} position, the vector $K_{ij} \cdot X_i$ is added to the total speed of the spaceship. Here, each K_{ij} is an integer known as transmission coefficient of j_{th} gear on i_{th} selector. Therefore, the total speed of the spaceship is an N-dimensional vector defined as (g_i is the position the i_{th} selector is set to)

$$S = \sum_{n=1}^{N} K_{ig_i} X_i$$

Wedge knows that, in order to make spaceship driving convenient, the directions and transmission coefficients of any spaceship, including this one, satisfy some additional constraints. First of all, the directions are chosen in such a way that for any desired N-dimensional speed vector S, it's possible to choose coefficients c_i such that

$$S = \sum_{n=1}^{N} c_i X_i$$

Note that c_i may or may not be equal to any of the K_{ij}.

Additionally, K_{ij} is always less than K_{it} if $j < t$. And the last but not least, for each selector, one of the transmission coefficients is equal to zero.

As we've already mentioned, the spaceship is now moving in some direction through the hyperspace, and Wedge doesn't know the directions and the transmission coefficients. He wants to stop the spaceship. In order to do that, he needs to find the positions for each selector such that the total speed S will be equal to zero. All he can do now is to set each selector to some position (that is, choose g_i for each i) and compute the resulting speed.

Your program has to play Wedge's role and choose positions of each selector. The jury's program will tell your program the resulting speed. Wedge is limited in time, so you could make no more than 120 queries.

Interaction Protocol

At first, your program will be given two integers N and M ($1 \le N, M \le 100$). After that, your program has to determine positions of each selector that will produce zero speed.

Though in real world the spaceship is very fast, its hyperspace speed is very limited. Wedge knows that all transmission coefficients and components of directions don't exceed 1000 by absolute value.

To make a query, the program must print the question mark ("?"), a space, and then N integers $g_1, g_2, ..., g_N$ separated by spaces. Each number must be in the range from 1 to M. This will mean that Wedge sets the first selector to position g_1, the second selector to position g_2, and so on. Then your program must read N integers: the coordinates of the resulting speed vector S.

When your program is ready to print the answer, it must print the English letter "A", a space, and then N integers $g_1, g_2, ..., g_N$ separated by spaces. After that, your program must exit immediately.

To prevent output buffering, flush the output buffer after each request: this can be done by using, for example, fflush (stdout) in C or C++, System.out.flush () in Java, flush (output) in Pascal or sys.stdout.flush () in Python.

Please note: if for some query, your program receives the number 987654321 as the first integer of the answer, it means that your solution is wrong, and your program must exit immediately. In that case, you can hope to receive the proper outcome ("Wrong Answer", "Presentation Error", etc.). If your program does not exit in such situation, you will most likely receive a random outcome instead of what actually happened.

Examples

standard input	standard output
2 2 0 -1 2 0 0 0	 ? 1 1 ? 2 2 ? 1 2 A 1 2
2 3 0 -2 200 0 0 -1	 ? 1 1 ? 3 3 ? 1 2 A 1 3

Hint

In the first example, there are two selectors and two positions for each of them. The directions are the following: $X_1 = (1, 0)$, $X_2 = (0, 1)$. The transmission coefficients for the first selector are $K_{11} = 0$, $K_{12} = 2$, and for the second selector, they are $K_{21} = -1$, $K_{22} = 0$. Therefore, to achieve zero speed, Wedge needs to set the first selector in the first position, and the second selector in the second position.

In the second example, there are also two selectors, but now, each of them has three different positions. The directions and transmission coefficients are the following:

$X_1 = (1, 0)$, $X_2 = (0, 1)$;

$K_{11} = 0$, $K_{12} = 100$, $K_{13} = 200$;

$K_{21} = -2$, $K_{22} = -1$, $K_{23} = 0$.

Problem H. Tree Destruction

Input file:	standard input
Output file:	standard output
Time limit:	2000 ms
Memory limit:	256 MB

You are given an unweighted tree with n vertices. Then $n - 1$ following operations are applied to the tree. A single operation consists of the following steps:

1.choose two leaves;

2.add the length of the simple path between them to the answer;

3.remove one of the chosen leaves from the tree.

Initial answer (before applying operations) is 0. Obviously after $n - 1$ such operations the tree will consist of a single vertex.

Calculate the maximal possible answer you can achieve, and construct a sequence of operations that allows you to achieve this answer!

Input

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

Next $n - 1$ lines describe the edges of the tree in form a_i, b_i $(1 \le a_i, b_i \le n, a_i \ne b_i)$. It is guaranteed that given graph is a tree.

Output

In the first line print one integer number — maximal possible answer.

In the next $n - 1$ lines print the operations in order of their applying in format a_i, b_i, c_i, where a_i, b_i — pair of the leaves that are chosen in the current operation $(1 \le a_i, b_i \le n)$, $c_i (1 \le c_i \le n, c_i = a_i \text{ or } c_i = b_i)$ — chosen leaf that is removed from the tree in the current operation.

See the examples for better understanding.

Examples

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

Problem I. Crazy Town

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1000 ms
Memory limit:	256 MB

Crazy Town is a plane on which there are n infinite line roads. Each road is defined by the equation $a_i x + b_i y + c_i = 0$, where ai and bi are not both equal to the zero. The roads divide the plane into connected regions, possibly of infinite space. Let's call each such region a block. We define an intersection as the point where at least two different roads intersect.

Your home is located in one of the blocks. Today you need to get to the University, also located in some block. In one step you can move from one block to another, if the length of their common border is nonzero (in particular, this means that if the blocks are adjacent to one intersection, but have no shared nonzero boundary segment, then it are not allowed to move from one to another one in one step).

Determine what is the minimum number of steps you have to perform to get to the block containing the university. It is guaranteed that neither your home nor the university is located on the road.

Input

The first line contains two space-separated integers x_1, y_1 ($-10^6 \le x_1, y_1 \le 10^6$) —the coordinates of your home.

The second line contains two integers separated by a space x_2, y_2 ($-10^6 \le x_2, y_2 \le 10^6$) —the coordinates of the university you are studying at.

The third line contains an integer n ($1 \le n \le 300$) —the number of roads in the city. The following n lines contain 3 space-separated integers ($-10^6 \le a_i, b_i, c_i \le 10^6; |a_i| + |b_i| > 0$) —the coefficients of the line $a_i x + b_i y + c_i = 0$, defining the i_{th} road. It is guaranteed that no two roads are the same. In addition, neither your home nor the university lie on the road (i.e. they do not belong to any one of the lines).

Output

Output the answer to the problem.

Examples

standard input	standard output
1 1 -1 -1 2 0 1 0 1 0 0	2
1 1 -1 -1 3 1 0 0 0 1 0 1 1 -3	2

Problem J. Consanguine Calculations

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3000 ms
Memory limit:	256 MB

Every person's blood has 2 markers called ABO alleles. Each of the markers is represented by one of three letters: A, B, or O. This gives six possible combinations of these alleles that a person can have, each of them resulting in a particular ABO blood type for that person.

Combination	ABO Blood Type
AA	A
AB	AB
AO	A
BB	B
BO	B
OO	O

Likewise, every person has two alleles for the blood Rh factor, represented by the characters '+' and '-'. Someone who is "Rh positive" or 'Rh+' has at least one '+' allele, but could have two. Someone who is "Rh negative" always has two '-' alleles. The blood type of a person is a combination of ABO blood type and Rh factor. The blood type is written by suffixing the ABO blood type with the '+' or '-' representing the Rh factor. Examples include 'A+', 'AB-', and 'O-'. Blood types are inherited: each biological parent donates one ABO allele (randomly chosen from their two) and one Rh factor allele to their child. Therefore 2 ABO alleles and 2 Rh factor alleles of the parents determine the child's blood type. For example, if both parents of a child have blood type A-, then the child could have either type 'A-' or type 'O-' blood. A child of parents with blood types 'A+' and 'B+' could have any blood type. In this problem, you will be given the blood type of either both parents or one parent and a child; you will then determine the (possibly empty) set of blood types that might characterize the child or the other parent. Note: an uppercase letter 'O' is used in this problem to denote blood types, not a digit (zero).

Input

The input consists of multiple test cases. Each test case is on a single line in the format: the blood type of one parent, the blood type of the other parent, and finally the blood type of the child, except that the blood type of one parent or the child will be replaced by a question mark. To improve readability, whitespace may be included anywhere on the line except inside a single blood type specification. The last test case is followed by a line containing the letters 'E', 'N', and 'D' separated by whitespace.

Output

For each test case in the input, print the case number (beginning with 1) and the blood type of the parents and the child. If no blood type for a parent is possible, print 'IMPOSSIBLE'. If multiple blood types for parents or child are possible, print all possible values in a comma-separated list enclosed in curly braces. The order of the blood types inside the curly braces does not matter. The sample output illustrates multiple output formats. Your output format should be similar.

Examples

standard input	standard output
O+ O- ? O+ ? O- AB- AB+ ? AB+ ? O+ E N D	Case 1: O+ O- {O+, O-} Case 2: O+ {A-, A+, B-, B+, O-, O+} O- Case 3: AB- AB+ {A+, A-, B+, B-, AB+, AB-} Case 4: AB+ IMPOSSIBLE O+

Problem K. String Modification

Input file:	standard input
Output file:	standard output
Time limit:	2000 ms
Memory limit:	512 MB

Snuke received a string s as a new year present. Determine if he can convert it to his favorite string, t, by repeating the following operation zero or more times.

Operation: Choose a character from s, and insert another character right after the chosen character. The inserted character must be different from the chosen character.

For example, he can convert "abca" to "adbca" in a single operation by choosing the first 'a' and inserting a 'd' right after it. However, he can't convert "abca" to "aabca" in a similar way.

Input

First line of the input contains string s, second line contains string t. Both strings are composed of lowercase English letters, $1 \leq |s| \leq |t| \leq 5000$.

Output

Print "Yes" in case when Snuke can convert s to t, or "No" otherwise.

Example

standard input	standard output
snuke snukent	Yes
snuke ssnuke	No