Problem A. Anno Domini 2022

Time limit:	2 seconds
Memory limit:	512 megabytes

Soon we will celebrate New Year 2022, but what does this number mean? As you possibly know, this dating system was invented in AD 525 by Dionysius Exiguus. He chose the birth of Jesus Christ as the starting point of the Years of Our Lord (Anno Domini in Latin, AD for short). All the years before that were counted backwards as years Before Christ (BC for short).

An interesting detail of this dating system is that there is no year 0 — year 1 BC is immediately followed by AD 1. Because of that, sometimes it is quite tricky to find time difference between two dates if these dates belong to two different eras.

To simplify this task, please write a program that computes how many years passed between January 1st of two years given in the input.

Input

Two years are specified on two sequential input lines. Each year is specified in one of two forms:

1. as letters `AD`, followed by a space and a positive integer without leading zeroes in range 1..9999;
2. as a positive integer without leading zeroes in range 1..9999, followed by a space and letters `BC`.

The years may be specified in arbitrary order — the earlier year is not necessarily given first.

Output

The only line of the output must contain one integer: the number of years that passed between January 1st of the earlier year and January 1st of the later year.

Examples

standard input	standard output

`1 BC`
`AD 1`
	`1`

`AD 1`
`AD 2001`
	`2000`

`AD 2022`
`5508 BC`
	`7529`

Problem B. Boris and Berta

Time limit:	2 seconds
Memory limit:	512 megabytes

Boris is making a quest for his sister Berta. One of the tasks is to find a point on the map that is n meters to the north from their house. But it's too easy if n is specified directly. Boris decided to use miles and cables to specify the distance.

He found out that there are a lot of different miles: from a 500-meter Chinese mile (called *li*) up to a 11 299-meter Norwegian mile (called *mil*). And a cable length can be anywhere from 169 to 220 meters.

Boris decided to use an m-meter mile and a c-meter cable. Now he wants to represent the n-meter distance as "M miles and C cables" with non-negative integers M and C as precisely as possible — that is, he wants to minimize $|M \cdot m + C \cdot c - n|$. Help him!

Input

Three lines contain an integer each: n — the distance to represent, m — the chosen length of a mile, and c — the chosen length of a cable ($1 \le n \le 10^9$; $500 \le m \le 11\,299$; $169 \le c \le 220$). All values are given in meters.

Output

Print two non-negative integers M and C — the best approximation for the distance of n meters using the chosen mile and cable lengths. If there are multiple best approximations, print any of them.

Examples

standard input	standard output
1234 500 169	0 7
1700 500 200	1 6

Note

There are two correct answers to the second example test: "`1 6`" and "`3 1`".

Problem C. Clean Up!

Time limit:	2 seconds
Memory limit:	512 megabytes

Once Charlie decided to start a new life by deleting all files in his Downloads directory. It's easy to do that using `bash` shell! It has two useful features: the "`rm`" command, which removes all files given as arguments, and patterns, which are replaced with the list of files matching them before executing the command.

Charlie ran "`rm *`", but received an "`Argument list too long`" response. Unfortunately, after `bash` replaced "`*`" with the names of all files in the Downloads directory, it failed to run the command because it had too many arguments.

After some experiments, Charlie realized he can execute "`rm abc*`" to delete all files with names starting with "`abc`" if there are at most k such files. If more than k files match this pattern, none of them will be deleted. Of course, he can replace "`abc`" with any string.

Help Charlie to find the smallest number of "`rm`" commands needed to delete all files. Assume that he can only use the "`rm`" command as "`rm <prefix>*`", where `<prefix>` consists of lowercase English letters (and can be empty).

Input

The first line contains two integers n and k — the number of files to delete, and the maximum number of files that can be deleted by one "`rm`" command ($1 \le n, k \le 3 \cdot 10^5$).

Each of the next n lines contains a single string, denoting a file name. All file names are distinct, non-empty, and consist of lowercase English letters. The total length of all file names doesn't exceed $3 \cdot 10^5$.

Output

Print a single integer — the smallest number of "`rm`" commands needed to delete all files.

Examples

standard input	standard output
4 2 a abc abd b	2
4 2 d c ab a	2
5 3 please remove all these files	3

Note

In the first example test, Charlie can execute "`rm ab*`" to delete files "`abc`" and "`abd`", and then execute "`rm *`" to delete files "`a`" and "`b`". Note that he can't just run "`rm *`" immediately, because initially all four files match an empty prefix.

Problem D. Day Streak

Time limit:	4 seconds
Memory limit:	512 megabytes

Recently Deltaforces, a famous competitive programming website, added a lot of new visual information to user profiles. In particular, there is a maximum day streak — the maximum number of days in a row with at least one problem solved. You decided that the maximum day streak in your profile does not accurately represent your training efforts. So you came up with a thought — what if you could change the time zone in your profile to increase the maximum day streak?

Let's formalize this setting as follows. Suppose you have solved n problems, and the i-th problem was solved at time a_i. There are m time zones, numbered from 0 to $m - 1$. The default time zone is 0. If you decide to change your time zone to t, all solutions' timestamps increase by t: the problem solved at time a_i is now considered to be solved at time $a_i + t$, for all i simultaneously.

The problem solved at time x is considered to be solved on day number $\lfloor \frac{x}{m} \rfloor$. Here $\lfloor v \rfloor$ means v rounded down to the greatest integer less than or equal to v.

To display the maximum day streak, Deltaforces finds such l and r that you have solved at least one problem on each of days $l, l+1, \ldots, r$, and $r-l+1$ is as large as possible. Then your maximum day streak is shown as $r - l + 1$.

Find the maximum day streak you can achieve by selecting a time zone.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 2 \cdot 10^5$). Description of the test cases follows.

The first line of each test case contains two integers n and m — the number of solved problems and the number of time zones ($1 \le n \le 2 \cdot 10^5$; $1 \le m \le 10^9$). The second line contains n integers $a_1, a_2, \ldots, a_n$ — distinct timestamps of your solutions, in chronological order ($0 \le a_1 < a_2 < \cdots < a_n \le 10^9$).

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

Output

For each test case, print two integers s and t — the maximum day streak and any time zone that achieves it ($1 \le s \le n$; $0 \le t < m$).

Example

standard input	standard output

5
4 10
0 3 8 24
2 10
32 35
10 1
0 1 3 4 5 6 7 10 11 12
10 24
0 1 3 4 5 6 7 10 11 12
8 24
26 71 101 147 181 201 244 268
	3 2
	2 5
	5 0
	2 12
	4 15

Note

In the first example test case, when you select time zone 2, the timestamps of your solutions change to 2, 5, 10, and 26. It means the problems are now considered to be solved on days 0, 0, 1, and 2; that is a 3-day streak. Time zones 3, 4, and 5 yield the same answer.

In the second example test case, timestamps of your solutions are 37 and 40 in time zone 5, which corresponds to days 3 and 4. Time zones 6 and 7 also work.

In the third example test case, only one time zone exists, and your maximum day streak is 5.

In the fourth example test case, you have solved many problems but in a short period of time, and you can't obtain more than a 2-day streak.

Problem E. Extreme Problem

Time limit:	2 seconds
Memory limit:	512 megabytes

Many problems given in programming competitions are extreme in this or that regard. Examples include:

- a problem that is solved by doing lots of heavy math on paper and then by printing one well-known number, rounded to the number of digits given in the input file;
- a problem that spans more than four pages and requires you to write a simulation system that tracks multiple skills of several secret agents and chooses the best combination of them for each mission;
- a problem for which it is a proven fact that no correct solution will ever exist, but it was still given in a contest.

This time you are given a problem that is also extreme, in that it has only eight possible tests. And, of course, it will be about something extreme.

We consider functions of two integer variables, defined on the $[-10; 10] \times [-10; 10]$ square. It means that you can perform a call to $f(x, y)$ only if x and y are integers and $-10 \le x, y \le 10$. Such a function f is said to have a *local minimum* at (x, y) if the following statements hold simultaneously:

- $f(x, y) < f(x - 1, y)$;
- $f(x, y) < f(x + 1, y)$;
- $f(x, y) < f(x, y - 1)$;
- $f(x, y) < f(x, y + 1)$.

A *local maximum* is defined in a similar way, only the inequalities are reversed. A function f is said to have a *plateau* at (x, y) if at least one of the following statements holds:

- $f(x, y) = f(x - 1, y)$;
- $f(x, y) = f(x + 1, y)$;
- $f(x, y) = f(x, y - 1)$;
- $f(x, y) = f(x, y + 1)$.

Note that all the function invocations above must happen on the points from the function domain, that is, the $[-10; 10] \times [-10; 10]$ square. In particular, this means that a point on the boundary of this square **cannot be** a local maximum or a local minimum, but can still be a plateau.

You need to find a function which has — or does not have, depending on the information in the input:

- multiple local maxima;
- multiple local minima;
- some plateaus.

Note that "multiple" means "at least two". Also note that your function shall be defined in a specific way; see the Output section for details.

Input

The input consists of three lines.

- The first line starts with "`Multiple local maxima: `" and ends with either "`Yes`" or "`No`". This specifies whether your function shall or shall not have multiple local maxima.
- The second line starts with "`Multiple local minima: `" and ends with either "`Yes`" or "`No`", and similarly deals with local minima.
- The third line starts with "`Plateaus: `" and also ends with either "`Yes`" or "`No`". This specifies whether your function shall or shall not have plateaus.

Output

The output shall consist of one line that defines your function.

If no such function can be represented as per the format below, print "`No solution`".

Otherwise, print your function using the reverse Polish notation. Recall that the reverse Polish notation is a way to describe a function using some sort of a stack machine: it is a sequence of operations, some of which push values onto the stack, while some pull a few values from the top of the stack, perform some math and push the result back to the stack.

Your function shall contain at most 1000 tokens, separated by single spaces, where each token is one of the following.

- An integer constant ranging from `-9` to `+9`. This will push the corresponding number onto the stack.
- A variable, either `x` or `y`. This will push the value of that variable onto the stack.
- An operation, which can be `+`, `-`, `*`, or `^`. The asterisk means multiplication, whereas the `^` character means raising to the power. Each of these operations will take two numbers from the stack, apply the operation and push the result back to the stack. The evaluation order is such that "`9 5 -`" evaluates to 4, and similarly "`x 2 ^`" evaluates to x^2. As a special case, 0^0 evaluates to 1.

Note that whenever one of the following things happens:

- an operation attempts to take a number from an empty stack;
- the `^` operation attempts to raise something to a negative power;
- the result of an operation overflows the 32-bit signed integer;
- or at the end of the evaluation the size of the stack is not equal to one —

you receive a Wrong Answer outcome for the test where it happened.

Examples

standard input	standard output

`Multiple local maxima: No`
`Multiple local minima: No`
`Plateaus: No`
	`x 3 - 4 ^ y -5 + 2 ^ +`

`Multiple local maxima: No`
`Multiple local minima: No`
`Plateaus: Yes`
	`1`

Note

The example answer to the first test encodes the function $(x-3)^4 + (y+(-5))^2$. Note that it has no local maxima, no plateaus, and just one local minimum.

Problem F. First to Solve

Time limit:	5 seconds
Memory limit:	512 megabytes

The famous Forcedeltas Programming Contest features n contestants, m problems, and lasts for k minutes.

For each contestant i and each problem j, an integer $a_{i,j}$ is known. If $a_{i,j} = 0$, it means that contestant i can not solve problem j. Otherwise, it means that contestant i can solve problem j in exactly $a_{i,j}$ minutes.

All contestants will follow the same strategy. Specifically, each contestant will form a list of all problems they can solve, shuffle the list uniformly at random, and solve the problems in that order, until the list ends or the contest is over.

For example, if the list for contestant i looks like $j_1, j_2, \ldots$ after shuffling, then they will solve problem j_1 at minute a_{i,j_1}, problem j_2 at minute $a_{i,j_1} + a_{i,j_2}$, and so on. Note that no problem can be solved at minute $k + 1$ or later.

We'll say that contestant i gets the *First to Solve* award for problem j if no other contestant solves problem j strictly earlier. In particular, it means that multiple contestants can get the award for the same problem.

Find the expected number of awards each contestant will get, modulo 998 244 353 (see the Output section for details).

Input

The first line contains three integers n, m, and k — the number of contestants, the number of problems, and the length of the contest in minutes ($1 \le n \le 500$; $1 \le m \le 26$; $1 \le k \le 300$).

The i-th of the following n lines contains m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ ($0 \le a_{i,j} \le k$). The j-th of these integers denotes the number of minutes required for contestant i to solve problem j, or 0 if contestant i can not solve problem j.

Output

Print n integers — the expected number of awards contestants $1, 2, \ldots, n$ will get, modulo 998 244 353.

Formally, let $M = 998\,244\,353$. It can be shown that the expected number of awards can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Print the integer equal to $p \cdot q^{-1} \bmod M$. In other words, print such an integer x that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Example

standard input	standard output

5 3 60
30 0 0
40 20 0
30 60 0
0 0 0
60 60 1
	1
	1
	249561089
	0
	499122177

Note

In the example test, contestant 1 will always get the award for problem 1, contestant 2 will always get the award for problem 2, the expected number of awards contestant 3 will get is $\frac{3}{4}$, contestant 4 will never get any awards, and the expected number of awards contestant 5 will get is $\frac{1}{2}$.

Problem G. Grand Center

Time limit:	1 second
Memory limit:	512 megabytes

There is no generally accepted standard of calculating the center point of some territory. This fact is widely used to promote different places. You can always find a meaning for the word "center", according to which your city would be the center of the country! Gloria decided to develop one universal standard and find the true center.

Consider any point inside a convex polygon, and any direction. There is a unique segment that passes through the point, is parallel to the direction, and has both ends on the boundary of the polygon. The point divides the segment into two parts. The *direction imbalance* is defined as the ratio of the length of the bigger part to the length of the smaller part. The *imbalance* of the point is the maximum direction imbalance over all directions.

Gloria finds this value interesting, and wants to define the *Grand center* of a convex polygon as the point inside it which has the minimum possible imbalance. Help her to calculate the imbalance of the Grand center of a given polygon.

Input

The first line contains a single integer n — the number of vertices of the given convex polygon ($3 \le n \le 10\,000$).

The i-th of the following n lines contains two integers x_i and y_i — the coordinates of the i-th polygon vertex ($-10^5 \le x_i, y_i \le 10^5$). The x-axis runs from left to right, and the y-axis runs from bottom to top. The vertices are numbered in counterclockwise order. The polygon is strictly convex: all internal angles of the polygon are strictly smaller than π.

Output

Print one real number — the imbalance of the Grand center of the given polygon. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-6}.

Examples

standard input	standard output

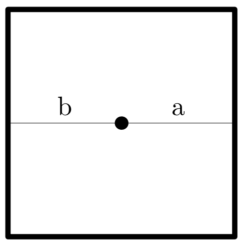
4
0 0
1 0
1 1
0 1
	1

3
0 0
1 0
0 1
	2

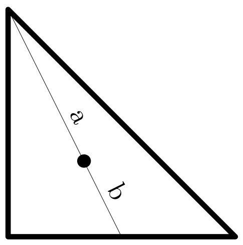
4
0 0
2 0
4 4
0 2
	1.5

Note

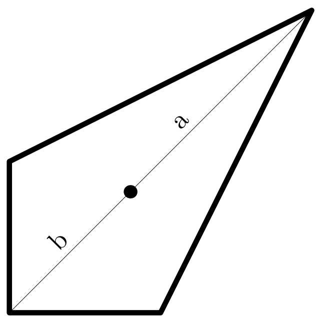
The following pictures illustrate all three example tests:



$a = 0.5$, $b = 0.5$
$a/b = 1$



$a \approx 0.744$, $b \approx 0.372$
$a/b = 2$



$a \approx 3.394$, $b \approx 2.263$
$a/b = 1.5$

In the first example test, for the center of the square, the direction imbalance is equal to 1 for any direction, i.e. any segment is split into two equal parts.

In the second example test, it's well-known that the median intersection point of a triangle splits the medians in a $2 : 1$ ratio. In the given triangle, the medians define the most imbalanced directions for that point. For any other point, the imbalance is even bigger.

In the third example test, the Grand center is located at $(1.6, 1.6)$.

Problem H. Halfway There

Time limit:	2 seconds
Memory limit:	512 megabytes

Given an integer n, find the median of the list of all integers from 1 to $n - 1$ that are coprime with n.

Recall that integers a and b are called *coprime* if their greatest common divisor is 1. The *median* of a list L is defined to be the $\frac{|L|}{2}$-th element of L if $|L|$ is even, and the $\frac{|L|+1}{2}$-th element of L if $|L|$ is odd. Here L is assumed to be sorted in ascending order, $|L|$ denotes the length of L, and indices are 1-based.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). Description of the test cases follows.

The only line of each test case contains a single integer n ($2 \le n \le 10^{18}$).

Output

For each test case, print a single integer — the median of the list of integers from 1 to $n - 1$ that are coprime with n.

Example

standard input	standard output
3 6 10 19	1 3 9

Problem I. Imprecise Permutation Sort

Time limit:	10 seconds
Memory limit:	512 megabytes

This is an interactive problem.

A permutation $a[1], a[2], \ldots, a[n]$ of integers from 1 to n is hidden from you.

Your task is to sort it in ascending order by comparing and swapping pairs of elements. This problem could be pretty easy, but the jury member responsible for the problem was too concentrated on floating-point arithmetic in problems G and J and implemented an "imprecise" comparator:

- if $\frac{|a[i]-a[j]|}{max(a[i],a[j])} \le 0.01$, then return 0;
- otherwise, if $a[i] < a[j]$, then return -1;
- otherwise, return 1.

Your program can make queries to compare any two elements with this comparator, or to swap any two elements. After each swap, it will be told whether the permutation became sorted.

Sort a permutation of size up to 16 384 using no more than 300 000 queries.

Interaction Protocol

Receive an integer n from the jury's program — the size of the permutation ($1 \le n \le 16\,384$). Then print queries and receive responses from the jury's program. After each query the output should be flushed and then a single integer should be read — the response to that query.

Comparison queries have a format "`C i j`", and swap queries have a format "`S i j`", where i and j are indices of two elements ($1 \le i, j \le n$). Making queries with $i = j$ is allowed.

The response to a comparison query is 0 if $a[i]$ "approximately equals" $a[j]$, otherwise: -1 if $a[i] < a[j]$, and 1 if $a[i] > a[j]$.

A swap query swaps values in $a[i]$ and $a[j]$, and the response to a swap query is 1 if after this swap the array became sorted in ascending order, and 0 otherwise.

Your program should exit as soon as it receives 1 as a response to a swap query.

The program should make at least one swap query. For example, if the hidden permutation is already sorted, it can make a query "`S 1 1`", receive a 1, and exit.

The interactor is not adaptive. The initial permutation is chosen by the jury's program in advance, before you make your first query.

Examples

standard input	standard output
4 -1 -1 1 1	 C 1 2 C 2 3 C 3 4 S 3 4
1 1	 S 1 1

Problem J. Journey in Fog

Time limit:	2 seconds
Memory limit:	512 megabytes

Julia and Jane are two friends living at the opposite ends of a long narrow street of length L.

Today, Julia needs to meet Jane and return home as soon as possible.

Jane has a list of speeds $v_1, v_2, \ldots, v_n$. At time 0, Jane picks an integer i from 1 to n uniformly at random, and starts moving towards Julia at a constant speed of v_i.

Julia is not as restricted in her movements, though. Starting from time 0, Julia can freely move along the street in any direction at any speed not exceeding V. In particular, Julia can stay at the same place as long as she wants, move at speeds lower than V, and change her speed at any moment.

It's foggy outside. Hence, Julia and Jane can not see each other unless they are at the same point of the street. Also note that Julia does not know Jane's speed, but she knows the list $v_1, v_2, \ldots, v_n$.

Suppose Julia meets Jane and arrives back home at time t. Julia will follow a strategy that minimizes the expected value of t. Find that expected value.

Input

The first line contains three integers n, L, and V — the number of speeds on Jane's list, the length of the street, and Julia's maximum speed ($1 \le n \le 10^5$; $1 \le L \le 10^9$; $1 \le V \le 10^6$).

The second line contains n integers $v_1, v_2, \ldots, v_n$ — the list of possible speeds of Jane in ascending order ($1 \le v_1 < v_2 < \cdots < v_n \le 10^6$).

Output

Print a single real number — the expected amount of time it will take Julia to meet Jane and return back home, if she follows an optimal strategy. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-9}.

Examples

standard input	standard output

`1 1000 30`
`10`
	`50.0000000000000`

`1 1000 10`
`30`
	`33.3333333333333`

`4 1000 20`
`10 20 30 40`
	`46.2500000000000`

Note

In the first example test, Julia is much faster than Jane. It's best for Julia to move towards Jane as fast as she can, meet her at time 25 at distance 750 away from home, and return back home at time 50.

In the second example test, Jane is much faster than Julia. It's best for Julia to just wait for Jane at home, where Jane will arrive at time $\frac{1000}{30}$.

Problem K. Kaleidoscopic Route

Time limit:	2 seconds
Memory limit:	512 megabytes

There are n cities in Kaleidostan connected with m bidirectional roads. The cities are numbered from 1 to n. Each road has an integer called *colorfulness*.

Keanu wants to travel from city 1 to city n. He wants to take the *shortest* route — that is, the route with the smallest number of roads. Among all the shortest routes, he wants to take the *kaleidoscopic* one — that is, the route with the largest possible difference between the maximum and the minimum colorfulnesses of its roads. Help Keanu find such a route.

Input

The first line contains two integers n and m — the number of cities and the number of roads ($2 \le n \le 10^5$; $1 \le m \le 2 \cdot 10^5$).

The i-th of the following m lines contains three integers v_i, u_i, and c_i — the indices of the cities connected by the i-th road, and the colorfulness of the i-th road ($1 \le v_i, u_i \le n$; $v_i \ne u_i$; $0 \le c_i \le 10^9$).

Each pair of cities is connected by at most one road. It is guaranteed that you can travel from any city to any other city using the roads.

Output

In the first line, print a single integer k — the number of roads in the required route.

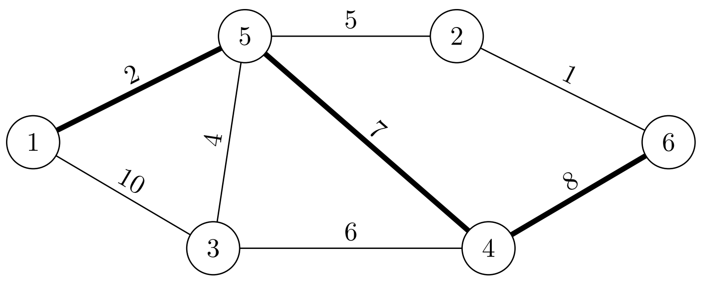
In the second line, print $k+1$ integers $c_0, c_1, \ldots, c_k$ — the sequence of cities on the route ($1 \le c_i \le n$; $c_0 = 1$; $c_k = n$).

Example

standard input	standard output

6 8
1 5 2
5 2 5
3 5 4
1 3 10
3 4 6
4 5 7
4 6 8
2 6 1
	3
	1 5 4 6

Note

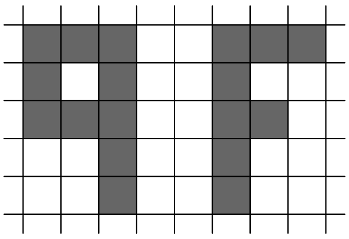


In the example test, the required route consists of 3 roads, and the difference between the maximum and the minimum colorfulnesses of its roads is $8 - 2 = 6$.

Problem L. Letters Q and F

Time limit:	2 seconds
Memory limit:	512 megabytes

Little Lev is learning how to draw letters `Q` and `F`. Initially, he has a white grid of size $n \times m$. Then he will draw several letters of one of the following two shapes:



Lev will not rotate or mirror these two shapes. Every time he draws a new letter, he will choose a position for the letter inside the grid and paint all cells of the shape black. Lev will only draw letters in such a way that before drawing all black cells of the letter are white — that is, he will never paint a cell twice.

You are given the final coloring of the grid. Count the number of letters `Q` and letters `F` drawn by Lev.

Input

The first line contains two integers n and m — the height and the width of the grid ($5 \le n \le 300$; $3 \le m \le 300$).

The next n lines contain m characters each, denoting the final state of the grid. A white cell is denoted by '`.`', a black cell is denoted by '`#`'.

It is guaranteed that the grid is a valid result of Lev's drawing.

Output

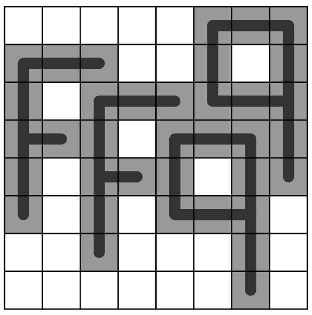
Print two integers — the number of letters `Q` and the number of letters `F` drawn by Lev, respectively.

Examples

standard input	standard output
5 3 ### #.# ### ..# ..#	1 0
5 3 ### #.. ##. #.. #..	0 1
5 8 ###..### #.#..#.. ###..##. ..#..#.. ..#..#..	1 1
8 8 .....### ###..#.# #.###### ###.#### #.###.## #.#.###. ..#...#. ......#.	2 2

Note

Illustration for the fourth example test:



Problem M. Multithreaded Program

Time limit:	2 seconds
Memory limit:	512 megabytes

Maurice is debugging a multithreaded program on his old machine. The program has several threads operating on a set of shared variables. Each thread executes its own sequence of assignments in a predefined *program order*. Each assignment sets one of the variables to an integer value.

When the program is run, assignments from different threads can be executed in any order. The only guarantee is that each thread executes all of its assignments in the program order.

For example, let's say the program has three threads that have 2, 2, and 1 assignments in their sequences, respectively. Then one valid program execution looks as follows:

- thread 1 executes the first assignment in its sequence;
- thread 2 executes the first assignment in its sequence;
- thread 2 executes the second assignment in its sequence;
- thread 1 executes the second assignment in its sequence;
- thread 3 executes the only assignment in its sequence.

This execution can be described as $1, 2, 2, 1, 3$, where numbers specify the threads performing each assignment, in order. Note that many other valid executions are possible.

Maurice suspects that his machine is broken and can work incorrectly. He has run his program and recorded the values of all variables at the end.

Find an execution of the program that performs all assignments and leads to the recorded values of all variables, or report that the machine is indeed broken and such an execution does not exist.

Input

The first line contains a single integer t — the number of threads ($1 \leq t \leq 100$). The threads are numbered from 1 to t. The following lines describe t sequences of assignments, one per thread.

The first line of the i-th description contains an integer l_i — the length of the sequence of assignments of the i-th thread ($1 \leq l_i \leq 100$). Each of the following l_i lines contains an assignment in the form "`<variable>=<value>`". The assignments are listed in the program order. Variable names consist of up to 10 lowercase English letters, and values are positive integers not exceeding 10^9.

The first of the remaining lines contains an integer k — the number of variables ($1 \leq k \leq 10\,000$). Each of the following k lines contains a variable name and its recorded value, which is a positive integer not exceeding 10^9. Each variable used in the program is listed exactly once, and each listed variable is used in at least one assignment.

Output

Print "`Yes`" if an execution producing the recorded values exists, and "`No`" otherwise.

If an execution exists, print a line containing $s = l_1 + l_2 + \cdots + l_t$ integers $c_1, c_2, \ldots, c_s$, describing such an execution ($1 \leq c_i \leq t$). This specifies that the first assignment is performed by thread c_1, the second one is performed by thread c_2, and so on. Each thread performs its assignments in the program order. After the s-th assignment, each variable must have the recorded value. The i-th thread must appear in the description exactly l_i times.

Examples

standard input	standard output

```
2
2
a=1
b=2
2
b=1
a=2
2
a 1
b 1
```
	`No`

```
3
5
start=1
counter=1111
counter=10
counter=3333
finish=1
4
start=2
counter=20
counter=10
finish=2
3
start=3
qwerty=787788
finish=3
4
counter 10
start 1
finish 1
qwerty 787788
```
	```
	Yes
	2 3 3 2 1 1 3 1 1 2 2 1
	```

Problem N. New White-Black Tree

Time limit:	7 seconds
Memory limit:	512 megabytes

Naomi has learnt about Red-Black trees, now it's time to learn about White-Black trees. She is reading an algorithms book. Some pages contain pictures of trees, but the edges of the trees faded out through all these years. According to the text, each of these edges should be either white or black.

Naomi noticed that each vertex has two integers written beside it. She guessed that the first integer is the number of white edges incident to the vertex, and the second is the number of black edges incident to the vertex.

Naomi recreated all the pictures. Can you do that?

Input

The first line contains an integer t — the number of pictures to recreate ($1 \le t \le 3 \cdot 10^5$).

The following lines describe t pictures. Each description starts with a line containing an integer n — the number of vertices in the tree ($1 \le n \le 3 \cdot 10^5$).

The i-th of the following n lines of the picture description contains two integers w_i and b_i — two integers written beside the i-th vertex of the tree: the number of white and black edges incident to the i-th vertex ($0 \le w_i, b_i \le n - 1$).

It is guaranteed that the sum of n over all the pictures does not exceed $3 \cdot 10^5$.

Output

Print t blocks of output, the i-th of which should contain the information about recreating picture i.

In the first line of each block print "`No`" if there is no way, and "`Yes`" if there is at least one way to recreate the picture. If there is a way to recreate the picture of the tree, print additional $n - 1$ lines, each of them containing two integers and a letter '`W`' for white or '`B`' for black: v_i, u_i and c_i, defining an edge between vertices v_i and u_i of color c_i ($1 \le v_i, u_i \le n$; c_i is either '`W`' or '`B`').

If there are multiple ways to recreate a picture, you can print any of them. The edges of the tree can be printed in any order.

Example

standard input	standard output

6
4
1 1
1 1
1 0
1 0
4
1 0
2 1
1 1
1 0
1
0 0
2
0 1
0 1
2
1 0
0 1
3
2 0
0 1
0 1
	Yes
	1 4 W
	2 3 W
	2 1 B
	No
	Yes
	Yes
	2 1 B
	No
	No