Problem A. Lily

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

- They serve the purpose of changing hydrogen into breathable oxygen, and they're as necessary here, as the air is on earth.
- But I still say, they're flowers.
- If you like.
- Do you sell them?
- I'm afraid not.
- But, maybe we can make a deal.
- What do you mean?
- Oh you see, you don't have to send them anywhere. I'll pay for them. But then, I'll leave them here, for you.

— *Assignment Outerspace,* 1960

Everything that has a beginning has an ending. My journey has been reaching its ending, and I've been ready to say goodbye to my yesterday. But you, my dear friend, your journey is still thriving here at the 2022 CCPC Guilin Site. We sincerely hope you find a brand new milestone here, and forge ahead in the future with your love and passion.

Lily means a kind of beautiful flower. She usually only blooms once a year, so it could be very precious if you see a lily blooming. However, she is highly toxic to cats, so you must be aware of keeping curious cats away from lovely lilies.

You have n grids of soil land in a row, for 1 to n, some with lilies blooming. We don't want to hurt lilies, as well as cats. You can put some cat food on the grids, but for any grid i with cat food, grids with indices falling in the range $[i-1, i+1]$ must not contain lily flowers. You love cats and lilies, so you want to maximize the number of grids having cat food.

Design a plan to fulfill the above requirements.

Input

There's a single integer n ($1 \le n \le 1\,000$) in the first line, denoting the number of grids.

The second line contains a string R consisting of only '`L`' and '`.`', denoting the grids with and without lilies.

Output

The output contains a single line with string R' consisting of only '`L`', '`.`' and '`C`', where '`C`' means the cat food you assigned to the empty grids in R while fulfilling the above requirements.

If there are multiple solutions, print any.

Examples

standard input	standard output
5 ..L..	C.L.C
2 ..	CC

Problem B. Code With No Forces

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Preparing a reliable programming contest is tricky work. Sometimes, you can make the test set huge (e.g., `wrong answer on test 400`) instead of guessing how participants will solve the problems. However, huge test sets could be a critical burden to the poorly implemented contest platforms. If the judging process is not parallelled well, the flow time of submission is unacceptable, and you will receive countless complaints about it.

The problem you've made contains n test cases, and there are m test solutions written by different testers that you can believe are the most typical solutions. To avoid long judging processes, you want to prune the test set by deleting some test cases and reordering the remaining if needed. After that, it's required to keep the results of every test solution remain unchanged. Your goal is to keep the least number of tests at last.

Take the following snapshot from the contest preparing platform, Polygon, as an example.

#	aho_tle.cpp	hash_1e18.cpp	hash_1e9.cpp	hash_kmp.cpp	hash_overflow.cpp	std.cpp
1	OK 0 / 34	OK 15 / 1	OK 0 / 1	OK 0 / 4	OK 0 / 1	OK 0 / 36
2	OK 0 / 34	OK 0 / 1	OK 15 / 1	OK 15 / 4	OK 0 / 1	OK 0 / 36
3	OK 0 / 34	OK 15 / 1	OK 0 / 1	OK 0 / 4	OK 15 / 1	OK 0 / 36
4	OK 0 / 34	OK 0 / 1	OK 0 / 1	OK 0 / 4	OK 15 / 1	OK 0 / 36
5	OK 0 / 34	OK 0 / 1	OK 15 / 1	OK 0 / 4	OK 0 / 1	OK 0 / 36
6	OK 0 / 34	OK 0 / 1	OK 0 / 1	OK 0 / 4	OK 0 / 1	OK 0 / 36
7	OK 0 / 34	OK 0 / 1	OK 0 / 1	OK 15 / 4	OK 0 / 1	OK 0 / 36
8	OK 46 / 34	OK 78 / 1	OK 78 / 1	OK 234 / 4	OK 30 / 1	OK 31 / 36
9	OK 31 / 35	OK 592 / 1	OK 499 / 1	OK 389 / 4	OK 390 / 1	OK 46 / 36
10	OK 93 / 34	OK 265 / 1	OK 217 / 1	OK 249 / 4	WA 140 / 1	OK 77 / 36
11	OK 93 / 34	OK 265 / 1	OK 233 / 1	OK 249 / 4	WA 140 / 1	OK 62 / 36
12	OK 779 / 34	OK 2932 / 1	OK 2433 / 1	OK 1730 / 399	OK 2089 / 1	OK 561 / 36
13	OK 811 / 34	OK 3119 / 1	OK 2589 / 1	OK 1809 / 397	OK 2245 / 1	OK 514 / 36
14	OK 811 / 34	OK 2964 / 1	OK 2308 / 1	OK 1731 / 390	OK 2028 / 1	OK 483 / 36
15	OK 109 / 35	OK 296 / 4	OK 249 / 4	OK 312 / 6	OK 140 / 3	OK 78 / 36
16	OK 78 / 35	OK 389 / 2	OK 373 / 2	OK 358 / 5	OK 280 / 2	OK 93 / 36
17	TL 5000 / 34	OK 140 / 1	OK 109 / 1	OK 280 / 6	OK 62 / 1	OK 78 / 36
18	OK 93 / 35	OK 171 / 3	OK 155 / 3	OK 280 / 6	OK 124 / 3	OK 124 / 36
19	OK 93 / 35	OK 202 / 3	OK 202 / 3	OK 327 / 6	OK 93 / 3	OK 109 / 36
20	OK 124 / 35	OK 124 / 2	WA 124 / 2	OK 265 / 5	OK 77 / 2	OK 109 / 36
21	OK 124 / 35	OK 155 / 2	WA 156 / 2	OK 249 / 5	OK 78 / 2	OK 140 / 36
Σ passed tests	20	21	19	21	19	21

Test solutions for *Last Warning Of Competition Finance Officer*, from the 2022 Shanghai Collegiate Programming Contest

The input data (See section **Input**) is given as the sheet above. For each solution, it will have a result with its verdict, running time and memory used on each test. The verdicts you should care about include the following, which are all common verdicts except **compile error**.

- `OK`: **correct**
- `WA`: **wrong answer**
- `TL`: **time limit exceeded**
- `ML`: **memory limit exceeded**
- `RE`: **runtime error**

After testing on a specific data set, the result including a representing verdict, a peak running time and a peak memory used will be returned to the participant. However, participants will not have information after the first failed test:

- When a solution passes all tests (**correct** on all test cases), the maximum time and memory will be shown with the **correct** verdict.
- Otherwise, the first failed test's verdict decides the verdict of the solution, and the maximum time and memory from the first test to the first failed test will be shown.

We can conclude from the above sheet that the verdicts of each solution are shown as follows.

Test Solution	Result
aho_tle.cpp	**time limit exceeded** 5000ms/35MB
hash_1e18.cpp	**correct** 3119ms/4MB
hash_1e9.cpp	**wrong answer** 2589ms/4MB
hash_kmp.cpp	**correct** 1809ms/399MB
hash_overflow.cpp	**wrong answer** 390ms/1MB
std.cpp	**correct** 561ms/36MB

To reduce the size of the test set, you can delete some tests, reorder the remaining ones, and make the results (verdicts, time, memory) of all tester solutions unchanged. Take the above sheet as an example:

- You can keep cases $9, 17, 10, 15, 13, 20, 12$ to satisfy the requirement, and it can be shown to be an optimal solution.
- You can reorder the cases if you like to – $9, 17, 10, 15, 13, 12, 20$ is also valid. However, swapping 9 and 17 is not valid since the solution 'aho_tle.cpp' will have a smaller memory cost.

You are given a test result sheet. Find an optimal way to reduce the test set so that the number of tests remaining is minimum.

Note that if a program is not tested on any test case, the status is undefined and not equal to any valid result, instead of '**correct** 0ms/0MB' on some online judges.

Input

The first line contains two integers n and m ($1 \le n \le 400,\ 1 \le m \le 6$), denoting the size of the test set and the number of tester solutions.

In the following n lines, the i-th line contains m strings separated by spaces, denoting the verdicts of the m test solutions of the i-th tests. The string is in the format '`{verdict},{time cost}/{memory cost}`', where the time cost and memory cost are integers, in milliseconds and megabytes, respectively.

It's guaranteed that:

- `verdict` $\in$ {`OK`, `WA`, `TL`, `ML`, `RE`}.
- $0 \le$ `time cost`, `memory cost` $\le 10\,000$.
- All results with `verdict` = `TL` has an equal maximum value of `time cost` that is higher than those with different verdicts.
- All results with `verdict` = `ML` has an equal maximum value of `memory cost` that is higher than those with different verdicts.

Output

In the first line, print a single integer $|S|$ denoting the minimum size of the test set satisfying the requirements mentioned above. Note that for a correct answer, $1 \le |S| \le n$ must hold.

In the second line, print $|S|$ integers separated by spaces, denoting the indices sequence S of tests that the new test set you constructed is in such order.

Examples

standard input	standard output
2 3 OK,1/1 OK,2/1 OK,2/2 WA,1/1 OK,1/1 TL,1000/1	2 1 2
3 3 OK,1/1 OK,2/1 OK,1/2 OK,3/3 OK,1/2 OK,114/514 WA,999/999 TL,3000/2 ML,999/1024	1 3
5 3 OK,0/0 OK,0/0 OK,0/0 WA,1/0 RE,0/0 OK,0/0 WA,0/0 WA,0/0 WA,0/0 OK,1/0 RE,0/0 OK,0/0 WA,2/2 RE,2/2 WA,2/2	2 4 3

Note

In sample case 3, you can find that the **runtime error** is reported by test 4 instead of 2 originally, but it's fine as long as the verdict is the same. The fifth test case is not useful even with corresponding verdicts on every solution, because it has a higher memory and time than the final result of each solution.

Problem C. Array Concatenation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Little relyt871 has a magical machine. In each operation, his machine can do one of the following operations to the input array b:

- Generate a copy of b and concatenate it after b. More formally, the resulting array should be
$$b' = \{b_1, b_2, \ldots, b_{|b|}, b_1, b_2, \ldots, b_{|b|}\}.$$
- Generate a copy of b, reverse it, then concatenate it before b. More formally, the resulting array should be
$$b' = \{b_{|b|}, b_{|b-1|}, \ldots, b_1, b_1, b_2, \ldots, b_{|b|}\}.$$

Initially, he has an array a of length n. Then, he wants to operate the machine exactly m times using the array on his hand while maximizing the sum of all prefix sums of the final array. Since he has a somewhat finite brain, when he adds some integers, he only cares about the sum modulo $1\,000\,000\,007$. Formally, suppose after all m operations he has array b of length n', he wants to maximize the following value:

$$\Big(\sum_{i=1}^{n'}\sum_{j=1}^{i} b_j\Big) \pmod{1\,000\,000\,007}.$$

Please note that you should maximize the value **after** taking the modulo: the array with answer $1\,000\,000\,007$ before taking the modulo is considered less than the array with answer 1.

Input

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

The second line contains n integers $a_1,\ a_2, ...,\ a_n$ ($1 \le a_i \le 10^9$) separated by spaces.

Output

Print a single integer in one line, denoting the answer.

Examples

standard input	standard output
2 1 1 2	15
5 10 26463 39326 86411 75307 85926	806275469

Problem D. Alice's Dolls

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Alice Margatroid is a skilled magician living in the Forest of Magic, who is good at making and controlling dolls. Today she is selling her self-made dolls to earn some money.

To attract people, Alice does not sell her dolls directly. Instead, she made a sweepstake (lottery) with different types of dolls. For each draw, the customer will get a special doll with a fixed probability $\frac{a}{b}$, a rational number within $(0, 1]$ (including 1 but not 0). Otherwise, the customer will get a doll of the usual kind.

By coincidence, Cirno goes to the Forest of Magic today. She finds the special dolls very attractive, but the usual dolls are not. Cirno decides to keep drawing until she gets n special dolls, but she does not have much money, so she wants to estimate how many times she needs to draw.

Given a non-negative integer m, suppose the times Cirno needs to draw is x. For each non-negative integer k between 0 and m (both inclusive), Cirno wants to know the expected value of x^k. The answers can be proven to be rational numbers, and you only need to output the answers modulo 998 244 353, which is a prime number.

The value of a rational number r modulo a prime p is defined as follows: suppose $r = \frac{x}{y}$, where x and y are non-negative integers, and $0 \le x < p$, $1 \le y < p$. Also suppose there is a non-negative integer $z \in [0, p-1]$ such that $yz \equiv 1 \pmod{p}$, then $r \equiv xz \pmod{p}$, in other words, the value of $r \bmod p$ is equal to $xz \bmod p$. It can be proven that there is a unique $z \in [0, p-1]$ for each y, so the value is surely determined.

Input

The only line of input contains four integers n ($1 \le n \le 10^5$), m ($0 \le m \le 10^5$), a, b ($1 \le a \le b \le 998\,244\,352$), indicating that Cirno wants to get n special dolls, and the upper bound of k is m (inclusive), and the probatility to get a special doll in a single draw is $\frac{a}{b}$.

Output

Output $m + 1$ integers separated with line breaks, the i-th ($0 \le i \le m$) of which represents the value of the expectation of x^i modulo 998 244 353.

Examples

standard input	standard output
1 3 1 2	1 2 6 26
100 5 1 6	1 600 363000 221433000 425510992 941092730

Problem E. Draw a triangle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Little Desprado2 is a student of Springfield Flowers Kindergarten. On this day, he had just learned how to draw triangles on grid coordinate paper. However, he soon found it very dull, so he came up with a more interesting question:

He had drawn two integral points of the triangle on the grid paper, and he denotes them (x_1, y_1) and (x_2, y_2). Now, he wanted to know the answer to the following question: where can he draw the third point (x_3, y_3) so that the area of the triangle is positive but minimized?

Obviously, he can't solve this problem because he is too young and simple. Can you tell him the answer?

Please note that your answer's coordinates must consist of integers because he is drawing on grid paper, and the triangle shouldn't be a degenerated triangle to keep the area positive.

Input

The first line contains one integer T ($1 \le T \le 50\,000$), denoting the number of Little Desprado2's queries.

For each test case, there's a single line contains four integers $x_1,\ y_1,\ x_2,\ y_2$ ($-10^9 \le x_1,\ y_1,\ x_2,\ y_2 \le 10^9$) seperated by spaces, denoting two points are at (x_1, y_1) and (x_2, y_2), respectively.

It is guaranteed that the two points won't coincide.

Output

For each test case, print two integers $x_3,\ y_3$ ($-10^{18} \le x_3,\ y_3 \le 10^{18}$) in a separated line, denoting your answer.

If there are multiple answers, you can print any one of them. It is guaranteed that there exists a solution in the above range.

Example

standard input	standard output
3 1 0 1 4 0 1 0 9 0 0 2 2	2 0 1 1 -1 0

Problem F. Union of Circular Sectors

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

You are given n non-degenerated circular sectors with the following guarantees:

- The central angle is less than or equal to 180 degrees.
- There are no two sectors with exactly the same radius and center.
- If two collinear radii belonging to two circular sectors exist, they won't share any point.

Please calculate the area of the union of those circular sectors.

Input

The first line contains one integer n ($1 \le n \le 1000$), denoting the number of semicircles.

Then follows n lines, the i-th line contains six integers $x_{i,o}$, $y_{i,o}$, $x_{i,a}$, $y_{i,a}$, $x_{i,b}$, $y_{i,b}$ ($-10^4 \le x_{i,o},\ y_{i,o},\ x_{i,a},\ y_{i,a},\ x_{i,b},\ y_{i,b} \le 10^4$), which denotes the coordinate of the center $O_i(x_{i,o},\ y_{i,o})$ and the two corners $A_i(x_{i,a},\ y_{i,a})$ and $B_i(x_{i,b},\ y_{i,b})$ of the i-th circular sector. The i-th sector is defined as the area that segment O_iA_i passes through when it rotates counter-clockwise around the point O_i to O_iB_i. It is guaranteed that:

- O_i, A_i and B_i won't coincide.
- $|O_iA_i| = |O_iB_i|$.
- $\overrightarrow{O_iA_i} \times \overrightarrow{O_iB_i} \ge 0$, where $\times$ denotes the modular of cross product of two 2-dimensional vectors, defined as $a \times b = a.x \cdot b.y - a.y \cdot b.x$. The equal sign is taken if and only if the two vectors are opposite.

Output

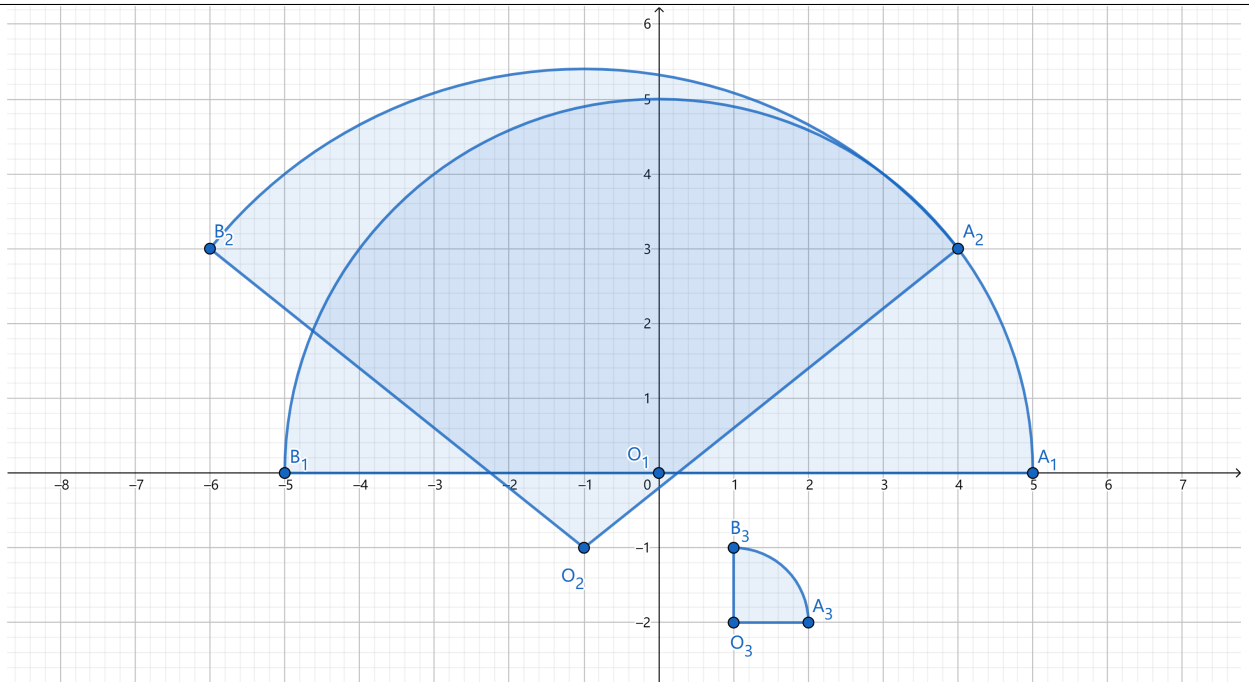
Print the area of the union of those circular sectors by a single decimal number in a single line. Your answer will be accepted if the absolute or relative error to the jury's answer is less than or equal to 10^{-6}.

Examples

standard input	standard output
3 0 0 5 0 -5 0 -1 -1 4 3 -6 3 1 -2 2 -2 1 -1	47.9378026054
1 0 0 -10000 -10000 10000 10000	314159265.3589793238

Note

The first test case can be illustrated in the following figure.


B2
A2
B1
O1
A1
O2
B3
A3
O3

Problem G. Group Homework

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

No, we don't want group homework. It's the place where $1+1<1$ can be true. However, you are currently the leader of a group with three members. Luckily, your group members will write everything down for you since you are the prestigious leader. Unluckily, you have to assign the new algorithm homework to your two team members, Mr. Dian and Mr. Beng, who can't understand the ideas of each other correctly and mess up all the details in the cooperation.

The new homework is about a tree: there are n vertices on the tree with $n-1$ bidirectional edges connecting them. Each vertex i is a problem with a score of a_i. You can assign a tree path to each member, then Mr. Dian and Mr. Beng will solve the problems on their path independently. The final score of the homework is decided by the sum of the scores of the vertices visited by **exactly one member** — as we mentioned before, if both of them solved one problem independently, they will argue a lot about who is right, and all the effort will be in vain.

Now, you — Mr. Ji — want to maximize the final score (as well as the chance not to drop out of school due to too low GPA). Make a wise choice!

Input

The first line of input contains a single integer $n\,(1 \le n \le 2 \times 10^5)$, denoting the number of vertices.

The second line contains n integers separated by spaces, the i-th integer denotes $a_i\,(1 \le a_i \le 10^4)$.

In the following $n-1$ lines, each contains two integers $u, v\,(1 \le u, v \le n)$, denoting (u, v) is a tree edge.

It is guaranteed that the input edges form a tree of n vertices.

Output

Print an interger in a single line, denoting the maximum score if assigning paths optimally.

Examples

standard input	standard output
5 1 9 9 9 9 1 2 1 3 1 4 1 5	36
1 1	0
3 1 2 3 1 2 2 3	6

Note

One optimal solution for sample case 1 is $(2, 3), (4, 5)$. Two paths will intersect at 1, and we can have all the remaining scores.

Note that you must assign exactly two paths to your dear group members. Therefore, for sample case 2, you can only assign $(1, 1), (1, 1)$ to Mr. Dian and Mr. Beng, which leads to a pretty 0 score.

Problem H. Hysteretic Racing

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

Racing is for high speed usually. However, sloths enjoy hysteretic racing. They believe being slower could be better for exhibiting racers' skills, and all sloths will drive at the same speed. The winner is the one with the best driving skill throughout the journey.

There is the best fit for the figure: the sloth racer and DMV officer called Flash, in the movie *Zootopia*. However, we use this AI-generated sloth driver instead to avoid legal issues.

The giant race track consists of n soil grids $0, 1, \ldots, n-1$ forming a circle, where the next grid of i is $(i+1) \bmod n$. The difficulty of the grid i is a **positive** integer d_i, and the grid with lowest difficulty, 1, is called a normal grid. The time cost to pass grid i is defined as follows: if the velocity v of sloths is higher than $\frac{1}{d_i}$ (normal grids per second) when entering grid i, the speed is adjusted to $\frac{1}{d_i}$ due to the laziness of sloths. After that, sloths will take d_i times more than the normal grid to pass it, i.e., it will cost $\frac{d_i}{v}$ seconds.

Sometimes, the sloths pay for diligent humans to rebuild the giant race track. There are two kinds of the job that will occur:

- **Plow** the soil in grids $l, (l+1) \bmod n, \ldots, r$, add a specific difficulty Δ to each grid.
- **Ram** the soil in grids $l, (l+1) \bmod n, \ldots, r$, replace each grid's difficulty to a specific difficulty d'.

To begin a racing game, the judge will choose a start grid s and the game's total time t seconds, and the game is organized in the following method:

- The sloths will have a velocity of 1 initially.
- At the beginning moment of the game, all sloths start entering the grid s.
- After going through a grid, sloths enter the next grid, where "next" is defined above.
- The game ends after t seconds. All sloths take a sharp brake and stop there.

Sloths can only walk very slowly, so the judge will immediately start to walk to the end position of the game. Now, you are also one of the paid humans who are to tell the judge where to go at the beginning of the game.

Since you are not a sloth, you have to do it fast.

Input

The first line of input contains $n, q\,(2 \le n, q \le 2 \times 10^5)$ separated by a single space, denoting the length of the giant race track and the number of operations and queries.

The second line contains n integer separated by spaces, the i-th denotes d_{i-1}, the inital difficulty of the i-th grid.

The following q lines contain queries and operations. Each line must fall in one of the three kinds:

- `P` l r Δ: **Plow** l-th to the r-th grid, with difficulty Δ.
- `R` l r d': **Ram** l-th to the r-th grid, with difficulty d'.
- `Q` s t: **Query** the end position of a game starting at $s\,(0 \le s < n)$ and with total time $t\,(0 \le t \le 2 \times 10^{17})$ seconds. Note that the position at the boundary of two grids is defined as the "next" grid, meaning that if slots exactly stop when leaving the i-th grid for the next, the position is defined as the grid $(i+1) \bmod n$.

It's guaranteed that $0 \le l, r < n$ for all l, r appeared in the q lines. We define the intervals $[l, r]$ precisely as follows:

- $l \le r : \{l, l+1, \ldots, r\}$;
- $l > r : \{l, l+1, \ldots, n-1, 0, 1, \ldots, r\}$.

It's also guaranteed that all difficulties appeared in the input, including d_i, Δ, d', are **positive** integers at most 10^6; the difficulties of grids after each operation are at most 10^6.

Output

In response to each query, you have to output a single integer in a single line denoting your answer.

Examples

standard input	standard output
2 4 1 2 Q 0 0 Q 0 1 Q 0 4 Q 0 5	0 1 1 0
5 7 2 5 1 4 3 Q 3 28 Q 0 39 P 4 0 3 Q 2 60 R 4 1 2 Q 1 22 Q 3 19260817	0 3 0 4 1

Note

In the first sample we illustrate how do we define the boundary. Starting a game at 0, it's considered in grid 0 for $t = 0$; it's considered in grid 1 for $t = 1$, since it only takes 1 second to pass the first grid. Then, the speed of sloths is reduced to $\frac{1}{2}$ afterwards, and it needs 4 seconds to pass grid 1.

Problem I. Invincible Hotwheels

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

Yukari is an elusive youkai (monster) who can manipulate boundaries. Recently Yukari is in a bad temper, and when she hears someone saying something she does not like (for example, "Yukari is thousands of years old"), she will be angry and beat the person from behind. Reimu has been bothered by her for a while because she does not know which words make Yukari unhappy. Even saying some common words like "coooooooooooooooooooooool", "I want to eat noodles", or "invincible hotwheels" will be beaten by Yukari.

To avoid angering Yukari, Reimu finds out n possible words Yukari may dislike, numbered from 1 to n. Each word is a string containing only lowercase English letters. However, some redundant (unnecessary) words may contain other words. For example, if "iwanttoeatnoodles" and "noodles" are both possible words, then the former is redundant.

Reimu wants to estimate how many redundant relations are among the n words. Formally, let s_i denotes the i-th word. Reimu wants to know that, how many tuples of $(i,\ j,\ k)$ $(1 \le i,\ j,\ k \le n;\ i, j, k$ are pairwise distinct) satisfying following contidions: s_i is a substring of s_j, and s_j is a substring of s_k. Also, there must not be another j' which is not equal to i, j or k, such that s_i is also a substring of $s_{j'}$, and $s_{j'}$ is also a substring of s_k.

Reimu asked you for help her calculating the number of such tuples.

Input

The first line of input contains one positive integer n $(1 \le n \le 10^6)$, denoting the number of words.

Then follows n lines. The i-th line contains a non-empty string s_i consisting of lowercase letters, denoting the i-th word. It is guaranteed that all the words are distince and $\sum_{i=1}^{n} |s_i| \le 2 \times 10^6$ holds.

Output

Output one integer - the answer described in the statement.

Examples

standard input	standard output
8 wwwsoupunetcom wwwsoupunet soupunet punetcom punet pun net n	6
4 a aa aaa aaaa	2
5 bc cbcb cbca cbc c	3

Note

For the first example, the valid tuples are $(3, 2, 1)$, $(5, 3, 2)$, $(6, 5, 3)$, $(6, 5, 4)$, $(7, 5, 3)$ and $(7, 5, 4)$.

Problem J. Permutation Puzzle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

Little relyt871 is solving a puzzle. The key to the puzzle is a permutation containing numbers $1 \dots n$. The values at some positions of the permutation are already fixed, and relyt871 needs to fill numbers into the remaining positions.

Besides, little relyt871 has gathered m extra requirements about the permutation. Let the solution be represented as $p_1, p_2, ..., p_n$, each clue is a pair of indices (u_i, v_i), which means that $p_{u_i} < p_{v_i}$ should be satisfied in the solution.

Little relyt871 wonders if all requirements may be satisfied at the same time. Write a program to find a valid solution when there is one.

Input

The first line of the input contains the number of test cases T ($1 \le T \le 20\,000$).

For each test case:

- The first line contains two integers n, m ($2 \le n \le 200\,000, 1 \le m \le 500\,000$).
- The second line contains n integers $p_1, p_2, ..., p_n$ ($0 \le p_i \le n$). If $1 \le p_i \le n$, then the value at position i is fixed as p_i, otherwise it is your task to determine the value at position i. It is guaranteed that for $1 \le i < j \le n$, if $p_i > 0$ and $p_j > 0$, then $p_i \ne p_j$.
- The following m lines each contains two integers u_i, v_i ($1 \le u_i, v_i \le n$), denoting the clues. It is guaranteed that the clues don't contradict themselves. Formally, there doesn't exist a list of clues $(u_{i_1}, v_{i_1}), (u_{i_2}, v_{i_2}), ..., (u_{i_k}, v_{i_k})$ such that $v_{i_j} = u_{i_{j+1}}, 1 \le j < k$ and $v_{i_k} = u_{i_1}$.

The sum of n over all test cases doesn't exceed 200 000, and the sum of m doesn't exceed 500 000.

Output

For each test case:

- If there exists no valid solution, output "`-1`" in a single line.
- Otherwise, output one line containing n integers seperated by spaces, denoting the solution. If there are multiple solutions, print any.

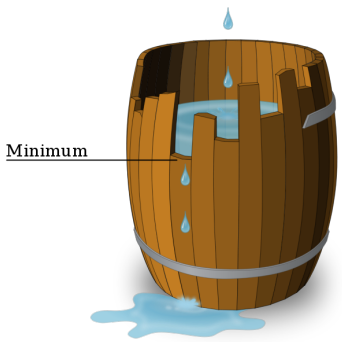
Examples

standard input	standard output
2 4 4 1 0 0 4 1 2 1 3 2 4 3 4 3 2 0 3 1 1 2 3 1	1 3 2 4 2 3 1
1 4 4 1 4 0 0 1 2 1 3 2 4 3 4	-1

Problem K. Barrel Theory

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Chinese children have more or less heard of the word "barrel theory" — the barrel's capacity is determined not by the longest wooden bars but by the shortest, as illustrated below.



Teachers and parents prefer to use this theory to tell the importance of all-round development, quality education, or collectivism. However, Little Desprado2 and Foolish Timsei disagree with this theory. In their opinion, a good barrel depends not only on large capacity but also on attractive looking.

According to barrel theory, if a barrel consists of n wooden bars of with positive integral length a_1, a_2, ..., a_n , the *capacity* of the barrel is $\min_{i=1}^{n} a_i$. They define the *ugliness* of the barrel is $a_1 \oplus a_2 \oplus ... \oplus a_n$. Here $\oplus$ denotes bitwise XOR (exclusive-or). They consider a barrel *good* if and only if its *ugliness* is **less than** its *capacity*.

Now Foolish Timsei and Little Desprado2 have a long wooden bar of length m, and they want to cut it into an n-pieces *good* barrel while the total length remains unchanged. Help them find a good scheme!

Input

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

Each test case contains two integers n and m ($1 \le n \le 10^5$, $n \le m \le 10^7$) in a single line, denoting the number of wooden bars after cutting and the length of the initial wooden bar.

It is guaranteed that sum of n over all test cases is not greater than 3×10^5; the sum of m over all test cases is not greater than 10^7.

Output

For each test case,

- If a cutting scheme exists, print "`YES`" in one line, followed by a line of n integers a_1, a_2, ..., a_n separated by spaces as the lengths of the n wooden bars. If there are multiple solutions, print any.
- Otherwise, print "`NO`" in one line.

Example

standard input	standard output
3 6 7 5 17 4 4	NO YES 2 2 2 4 7 YES 1 1 1 1

Problem L. Largest Unique Wins

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

n players are playing a game. They are to choose a number from $1, 2, \ldots, m$ at the same time, and the game result should be analyzed as follows:

- If there exists a number that exactly 1 player chooses, then the player with the largest such number wins and scores 1 point; the other players score -1 point.
- Otherwise, the game is declared no winner; everyone is rewarded with a 0 point.

The game plays only one round. To keep the game fair, the judge comes up with a new playing method for this game. Instead of giving a simple action, every player should submit their **strategy** to the judge, which should be a m-dimensional vector $s = (p_1, p_2, \ldots, p_m)$, where $\sum_{i=1}^{m} p_i = 1$ must hold. After that, the judge starts to roll the dice. For player i, the judge will sample a result uniformly at random according to s_i: with probability $s_{i,j}$, the number j is chosen. After sampling n numbers, the game concludes by the above rules.

The **Nash equilibrium**, named after the mathematician John Nash, is the most common way to define the solution of a non-cooperative game involving two or more players. In a Nash equilibrium, each player is assumed to know the equilibrium strategies of the other players, and no one has anything to gain by changing only one's own strategy. To be more precise, we can define the term **Nash Equilibrium** on this game: given n vectors of each player's **strategy**, if for each player, their strategy is the best response to the other players' strategies, we call it a Nash equilibrium. In other words, one can not earn a higher expected score in a Nash equilibrium by peeping at others' strategies and making some changes to their strategy.

Now, we'd like you to find a **Nash equilibrium** given n and m. Note that you don't have to maximize or minimize anything related to scores, but you should guarantee that no one can take advantage by changing strategy on the spot.

Input

The only input line contains two integers $n, m\,(2 \le n, m \le 12)$, denoting that it's a game that involves n players, and each player has to choose from $1 \ldots m$.

Output

Print n lines, each contains m decimal numbers. The the j-th decimal number $s_{i,j}$ on i-th line means that player i plays j with probability $s_{i,j}$.

You should guarantee the following precision requirements:

- $\forall i \in \{1, \ldots, n\}, \left|\sum_{j=1}^{m} s_{i,j} - 1\right| < 10^{-8}$.
- $\forall i \in \{1, \ldots, n\}$, replacing s_i to another **strategy** s_i' can not have a expected reward gain more than 10^{-8}.

Printing 10 or more digits after the decimal point is recommended to fulfill precision requirements.

Nash showed that there is always a Nash equilibrium for every finite game. If there are multiple solutions, print any.

Examples

standard input	standard output
2 2	0.0000000000 1.0000000000 0.0000000000 1.0000000000
3 3	0.2360679775 0.2360679775 0.5278640450 0.2360679775 0.2360679775 0.5278640450 0.2360679775 0.2360679775 0.5278640450
3 2	1.0000000000 0.0000000000 0.0000000000 1.0000000000 0.1145141919 0.8854858081

Note

The reference outputs for the first and the second sample are symmetric, meaning that every player uses the same strategy. We can prove that the reference output is the only Nash equilibrium in the first sample. However, asymmetric solutions exist for the second sample — even if players take different strategies, no one has the incentive to change their own given that the other players stick to their previous choice.

In the third sample case, we present an asymmetric example for you. The most interesting thing about it is viewing from the player 3's aspect. Fixing $1, 2$'s strategy,

- If player 3 chooses 1, player 2 wins.
- If player 3 chooses 2, player 1 wins.

Therefore, player 3's revenue is always the same, -1. However, 3's strategy can decide who wins for players 1 and 2. It can also be verified that player $1, 2$ have no incentive to change strategies too due to the existence of player 3.

Problem M. Youth Finale

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

Finales are born to be exciting. Performers play hard to draw audiences' attention and then take a perfect curtain call. As the last problem and the finale of the problem set, however, we want you to recall a simple algorithm. Like me, it may be the first algorithm you've learned, called Bubble Sort.

```
void bubble_sort(int a[], int n) { // 0-based, sort from lowest to highest
    for (int i = 1; i < n; i++) {
        for (int j = 0; j < n - i; j++) {
            if (a[j] > a[j + 1]) {
                swap(a[j], a[j + 1]);
            }
        } // after i-th inner iteration, a[n - i] is correct
    }
}
```

Given a permutation of length n, as you might know, Bubble Sort runs in $\Omega(n^2)$ in the worst case. It's quite a traditional idea to count the number of calls of "`swap`" in the algorithm. As you are stronger now, you want to count that number in a dynamic permutation with the following events that might happen:

- **Reverse** the permutation, meaning that the permutation is replaced with

$$p' = \{p_n, p_{n-1}, \ldots, p_2, p_1\}.$$

- **Shift** the permutation to the left by 1, meaning that the permutation is replaced with

$$p' = \{p_2, p_3, \ldots, p_n, p_1\}.$$

All you need to do is to output the number of "`swap`" that would be called if we sort the permutation with the above Bubble Sort code after each operation.

Input

The first line contains two integers $n, m (1 \le n \le 3 \times 10^5, 1 \le m \le 6 \times 10^5)$, denoting the length of permutation and the number of operations.

The second line contains n integers separated by spaces, and the i-th denotes the initial p_i.

The third line contains a single string containing m letters consisting of '`R`' and '`S`'. The i-th letter denotes the i-th operation, where '`R`' or '`S`' denotes the **Reverse** or **Shift** operation, respectively.

It's guaranteed that p forms a correct permutation of $1, 2, \ldots, n$.

Output

In the first line, print the number of "`swap`" would be called when Bubble Sort the initial p.

In the second line, print a single string of m digits. The i-th denotes the number of "`swap`" would be called to Bubble Sort the permutation, **modulo** 10.

Example

standard input	standard output
5 10 5 4 3 2 1 SSSSRSSSSR	10 6446466400