Problem A. Alphabet City

Time limit: 3 seconds
Memory limit: 1024 megabytes

Al is an urban designer in Alphabet City, and today their task is to prepare signs for n city streets. In Alphabet City, the street signs simply consist of the street names composed of capital identically stamped English metal letters. For instance, if, during nighttime, someone sneakily exchanges the first letters on NERC street and on NEF street, the next day nobody will see the difference as the letter 'N' is identical on both signs.

Al is planning to put m signs on each street, and they have already ordered the required number of brass letters for each of the street names from the metallurgical plant. However, one hour before the order arrived, Al got a call from the plant's manager with a devastating piece of news: they lost one of the items from the list of street names! To mitigate the issue, Al decided for now to put as many signs as possible on each street whose order was not lost, and, with the leftover letters, to prepare at least one sign for the street whose order was lost.

Formally, if $s_1, \ldots, s_n$ are the street names, and ℓ is the index of the missing item from the order, Al is interested in the maximum integer k such that it is possible, from all the letters of m copies of $s_1, \ldots, s_{\ell-1}, s_{\ell+1}, \ldots, s_n$, to compose k copies of $s_1, \ldots, s_{\ell-1}, s_{\ell+1}, \ldots, s_n$ and additionally at least one copy of s_ℓ, or to determine that such a composition is impossible for any non-negative k.

Al still does not know which of the items was lost. Write a program that, given m and all street names, for each $\ell \in \{1, 2, \ldots, n\}$ prints the maximum value of k, or -1 if such a composition is impossible.

Input

The first line consists of two integers n and m, denoting the number of streets in Alphabet City for which the signs are needed and the number of copies of each street name that Al initially ordered ($2 \le n \le 2 \cdot 10^5$; $1 \le m \le 5 \cdot 10^5$). Each of the next n lines consists of one string s_i — the street name ($1 \le |s_i| \le 5 \cdot 10^5$). All these names are composed of capital English letters. Some or all of these names *may* coincide. It is guaranteed that the sum of the lengths of all the street names does not exceed $5 \cdot 10^5$.

Output

Print n integers, the ℓ-th of them denoting the maximum integer k such that the letters of $m \times s_1$, …, $m \times s_{\ell-1}$, $m \times s_{\ell+1}$, …, $m \times s_n$ (where $m \times s$ denotes m copies of street name s) are enough to compose $k \times s_1$, …, $k \times s_{\ell-1}$, $1 \times s_\ell$, $k \times s_{\ell+1}$, …, $k \times s_n$. If, for the given value of ℓ, these letters are insufficient for any integer $k \ge 0$, print -1 instead.

Examples

standard input	standard output
3 10 NEERC NERC NEF	9 9 -1
4 4 LENSE TEN SENSELESSNESSES LENSE	3 -1 0 3

Problem B. Battle of Arrays

Time limit: 3 seconds
Memory limit: 1024 megabytes

Alice and Bob play a turn-based game. Initially, Alice has an array a of n positive integers, and Bob has an array b of m positive integers. The players take turns, with Alice moving first.

On a player's turn, they must choose one element x from their own array and one element y from their opponent's array. Then they perform the following operation:

- If $y \le x$: the element y is destroyed (removed from the opponent's array).
- If $y > x$: the element y is decreased by x (the value of y becomes $y - x$).

A player wins if, after their move, the opponent's array becomes empty.

Assuming both players play optimally, determine the winner.

Input

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

The first line of each test case contains two integers n and m ($1 \le n, m \le 10^5$) — the sizes of Alice's and Bob's arrays respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — Alice's array.

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le 10^9$) — Bob's array.

It is guaranteed that the sum of n over all test cases does not exceed 10^5 and the sum of m over all test cases does not exceed 10^5.

Output

For each test case, print the name of the winner of the game if both players follow the optimal strategy: "`Alice`" or "`Bob`".

Example

standard input	standard output
2 1 1 70 90 2 3 30 30 20 20 40	Alice Bob

Note

In the first test Alice moves and decreases Bob's element by 70, so it becomes 20. Then Bob moves and decreases Alice's element by 20, so it becomes 50. Finally, Alice moves, destroys Bob's element, and wins.

Problem C. Cacti Classification

Time limit:	3 seconds
Memory limit:	1024 megabytes

Ivan and Petr like to play with *cacti* — special graphs where each edge belongs to at most one simple cycle, and the graph is connected. Multiple edges between pairs of vertices and loops are allowed.

They invent the following game:

- Petr secretly builds a cactus with n vertices and m edges. The edges are labeled from 1 to m.
- Petr only tells Ivan the number m.
- Ivan is then allowed to ask questions of the following form:
  - He chooses a subset S of edge labels (see below about limitations on the subset), and asks: "If we only keep the edges whose labels are in S (and all n vertices), is the resulting graph connected?"
  - Petr must answer either "yes" or "no".

After asking at most $8m$ questions, Ivan must determine, for every edge:

1. whether this edge lies on some cycle in the cactus;
2. if it does, what is the length of that simple cycle.

In this problem, each loop is considered a simple cycle of length 1 and two edges between the same pair of vertices form a simple cycle of length 2.

However, Ivan is still very young and only knows numbers up to 14. So:

- if an edge lies on a simple cycle of length at most 14, he must output that exact length;
- if an edge lies on a simple cycle of length greater than 14, he must say that this edge lies on a *big cycle*.

Also, to avoid having to list a lot of edges each time, Ivan always asks about an edge set obtained from the set used in one of the previous queries, or from the set of all edges, by removing exactly one edge.

Can you design a strategy that allows Ivan to complete this task?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). The description of the test cases follows.

The first line of each test case contains m ($1 \le m \le 10^4$) — the number of edges in the cactus.

It is guaranteed that the sum of m over all test cases does not exceed 10^4.

Interaction Protocol

To ask a query, output a line in the following format (without the quotes):

- "? p e" ($0 \le p \le q$; $1 \le e \le m$), where p is the number of a previous query or zero (queries are numbered from 1 in the order they are asked), q is the number of queries made before the current one, e is the label an edge.

The query denotes a graph consisting of the edges used in the query number p (or all edges if $p = 0$), with edge e removed. The interactor checks whether this graph is connected when considering **all original vertices** and returns a single integer:

- 1 if the graph denoted by the query is connected;
- 0 if the graph denoted by the query is not connected;
- -1 if you have exceeded $8m$ allowed queries or edge e was already removed from the set of edges used in query p. In this case you should terminate your program to receive a Wrong Answer verdict.

When you have found the answer, output a single line in the following format:

- "! e_1 e_2 ... e_m" ($-1 \le e_i \le 14$ for all i),

where:

- if e_i is positive, then edge number i belongs to a simple cycle of length exactly e_i,
- if $e_i = 0$, then edge number i belongs to a big cycle (simple cycle of length greater than 14),
- if $e_i = -1$, then edge number i does not belong to any cycle.

The interactor returns a single integer:

- 1 if your answer is correct. Proceed to the next test case or terminate your program if this was the last case.
- -1 if your answer is incorrect. In this case you should terminate your program to receive a Wrong Answer verdict.

The interactor is **not** adaptive, meaning that the graph is fixed before you ask any queries.

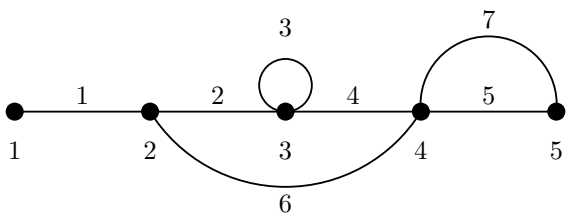
Example

standard input	standard output
1 7 0 1 1 1	 ? 0 1 ? 0 3 ? 2 4 ! -1 3 1 3 2 3 2

Note

In the example interaction, the input and output contain empty lines to align interactor responses with queries. These empty lines do not appear in the actual input and output.

In this example, the graph has 5 vertices and 7 edges; edges 1 through 7, in this order, are $(1, 2)$, $(2, 3)$, $(3, 3)$, $(3, 4)$, $(4, 5)$, $(2, 4)$, $(4, 5)$.



Problem D. Doorway

Time limit: 3 seconds
Memory limit: 1024 megabytes

The construction of the doorway for the Nonsense Engineering and Research Convention was delegated to one of the future attendees, who decided on a multi-layered sliding door design.

Each layer can be described as a horizontal interval, bounded by solid walls on the left and right, containing a number of sliding doors of fixed lengths. Within a layer, each door can move independently to the left or right, as long as it does not overlap other doors or the walls. All layers are parallel and stacked vertically.

After construction, the organizers noticed a problem: it is difficult to fully open the door, and since a large number of attendees are expected, they need to create the largest possible opening to allow everyone to pass through freely.

The size of the opening is defined as the total length of horizontal intervals such that, at every point of such an interval and in every layer, there is neither a door nor a wall. Your task is to determine the largest possible opening, given the doors' layout.

Input

The first line contains an integer n ($1 \le n \le 100\,000$) — the number of layers of the door.

Each of the next n lines starts with three integers k_i, $x_{i,1}$, $x_{i,2}$ ($0 \le k_i \le 300\,000$; $0 \le x_{i,1} < x_{i,2} \le 10^9$) — the number of sliding doors on that layer and the x-coordinates $x_{i,1}$ and $x_{i,2}$ of the walls on that layer. There is a wall at $x_{i,1}$ and a wall at $x_{i,2}$; all positions with $x < x_{i,1}$ or $x > x_{i,2}$ are blocked by walls.

They are followed by k_i integers $l_{i,1}, \ldots, l_{i,k_i}$ ($1 \le l_{i,j}$; $\sum_{j=1}^{k_i} l_{i,j} \le x_{i,2} - x_{i,1}$) — the lengths of the sliding doors on that layer given in order from the leftmost door to the rightmost.

It is guaranteed that $\sum_{i=1}^{n} k_i \le 300\,000$.

Output

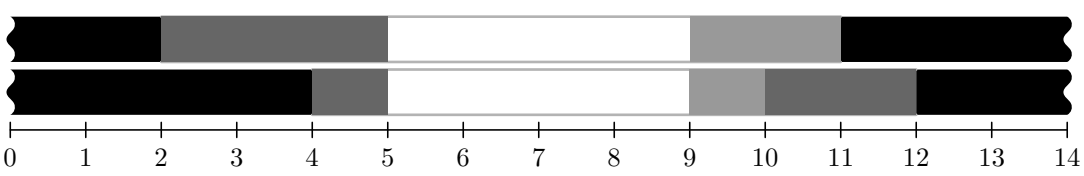
Output a single integer — the size of the largest possible opening that can be achieved by moving the sliding doors on each layer.

Examples

standard input	standard output
2 2 2 11 3 2 3 4 12 1 1 2	4
2 2 0 7 2 4 1 4 9 4	0

Note

This illustration shows a solution for the first example. Walls are filled with black color, doors are filled with various shades of grey, the opening is white. When first doors on each layer are shifted to the left and the rest of the doors to the right, we get the largest opening of 4.



Problem E. Elevator Against Humanity

Time limit: 3 seconds
Memory limit: 1024 megabytes

Nowadays, all gadgets are smart: smart phones, smart speakers, smart bulbs, and even smart elevators. The machines rise up.

The headquarters of the human resistance is located in a skyscraper. However, the smart elevator tries to slow people down without revealing itself.

There are n people in the skyscraper waiting for the elevator on different floors. Each person wants to get to another floor. Each person's destination floor is different from every other destination floor and from all starting floors. At the beginning, the elevator is located on the first floor. It moves one floor per unit of time. Whenever the doors open, it chooses the next floor and travels directly to it. The elevator can either go to the starting floor of the person who hasn't boarded the elevator yet and take them, or go to the destination floor of the person who is already in the elevator and disembark them. Note that the elevator doesn't stop on the intermediate floors even for the people who are already inside the elevator. The passenger boarding and disembarkation take negligible time. The elevator is big enough to accommodate all people at the same time.

The goal of the elevator is to maximize the total time until all passengers have been delivered to their destination floors. Find the maximum total time starting from the first floor until the disembarkation of the last passenger. Elevator doesn't need to return to the first floor.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10\,000$). The description of the test cases follows.

The first line of each test case contains a single integer n denoting the number of people ($1 \le n \le 10^5$).

Each of the following n lines contains two integers s_i and f_i ($2 \le s_i, f_i \le 10^9$) denoting the starting and the destination floor of the person i, respectively. All $2n$ floors in the input are pairwise distinct.

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

Output

For each test case, print the maximum time needed to transport all people.

Example

standard input	standard output
3 1 5 3 2 2 7 8 4 2 10 8 6 3	6 21 21

Note

In the first test case, there is only one person. The sequence of visited floors is $1 \to 5 \to 3$, and the time is 6.

In the second test case, one of the correct sequences is $1 \to 8 \to 2 \to 7 \to 4$ with the time 21.

In the third test case, one of the correct sequences is $1 \to 10 \to 6 \to 3 \to 8$ with the time 21.

Problem F. Fragmented Nim

Time limit: 3 seconds
Memory limit: 1024 megabytes

The classical game of Nim goes as follows. There are n piles of stones, and pile i initially consists of a_i stones. Alice and Bob take turns; Alice goes first. On their turn, a player chooses any non-empty pile and removes any positive number of stones from it. The player who takes the last stone wins.

After playing a lot of Nim, Alice and Bob decided to vary the rules a little bit. In this variation, the player whose turn it is does not choose a pile — *their opponent* does it for them! However, the player still gets to decide the number of stones to remove from that pile.

Alice still moves first. On Alice's turn Bob chooses any non-empty pile, and then Alice removes any positive number of stones from it. Similarly, on Bob's turn Alice chooses any non-empty pile, and then Bob removes any positive number of stones from it.

For the given configuration of stones in the piles, determine who will win if both players follow the optimal strategy.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n, denoting the number of piles ($1 \le n \le 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$, denoting the number of stones in the piles ($1 \le a_i \le 10^9$).

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 the name of the winner of the game if both players follow the optimal strategy: "`Alice`" or "`Bob`".

Example

standard input	standard output
3 3 1 2 3 1 1 5 10 3 4 7 4	Bob Alice Alice

Note

In the first test case, here's one possible way the game could proceed:

- Bob chooses the first pile. Alice removes 1 stone from it.
- Alice chooses the third pile. Bob removes 2 stones from it.
- Bob chooses the third pile. Alice removes 1 stone from it.
- Alice chooses the second pile. Bob removes 2 stones from it and wins.

In the second test case, Bob chooses the only pile, and Alice wins by removing the only stone from it.

Problem G. Greta's Game

Time limit: 3 seconds
Memory limit: 1024 megabytes

Greta and Alice are the two permanent hosts of the hit comedy show "QuestExpert". For this season they invited n programmers to complete quests, set by Alice. After that they all meet in a studio to review how well they did and complete the final studio quest.

Today, the studio quest that Alice came up with is as follows: first, all n participants stand in a circle in order from 1 to n counter-clockwise. Then Alice holds some number of rounds. In each round, every participant writes down an integer on a piece of paper. After that, Alice checks the numbers and for each i from 1 to n, if the i-th participant's number is strictly larger than the number of the next participant in counter-clockwise order (participant number $(i \bmod n) + 1$), then the i-th and the $(i \bmod n) + 1$-st participants both receive one point. After all rounds are complete, Alice calculates the total number of points for each participant and reports them to Greta. It turned out that the i-th participant scored a_i points.

Greta thinks that math games are boring, and this one took too long. To prove her wrong, Alice decides to cheat a little and instead of telling Greta the real number of rounds, she will tell her the minimum possible number of rounds that could still result in the i-th participant scoring a_i points for each i.

Help Alice determine this number.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n, denoting the number of participants ($2 \le n \le 5 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$, denoting the final scores of the participants ($0 \le a_i \le 10^9$). It is guaranteed that those scores were achieved in the described game with at least one round.

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

Output

For each test case, output on a separate line the minimum number of rounds that could lead to the given scores.

Example

standard input	standard output
5 2 3 3 3 2 2 2 4 1 2 4 3 5 0 2 3 5 4 6 5 8 3 10 14 4	3 2 2 4 10

Problem H. Honey Cake

Time limit: 3 seconds
Memory limit: 1024 megabytes

Hannah and Henry are going to host a party for n people, including themselves.

They bought a honey cake of size $w \times h \times d$ inches for the party, and want to split it into n equal pieces.

The honey cake can be cut parallel to any of its faces. To make cuts precise, each edge of length w is cut into the same number of equal parts, each having integer length; similarly for edges of lengths h and d.

Given the dimensions of the honey cake, determine whether it is possible to cut it into n equal pieces, and if so, how.

Input

The first line of input contains three integers: w, h, and d, the dimensions of the honey cake in inches ($1 \le w, h, d \le 10^9$).

The second line contains a single integer n ($1 \le n \le 10^9$).

Output

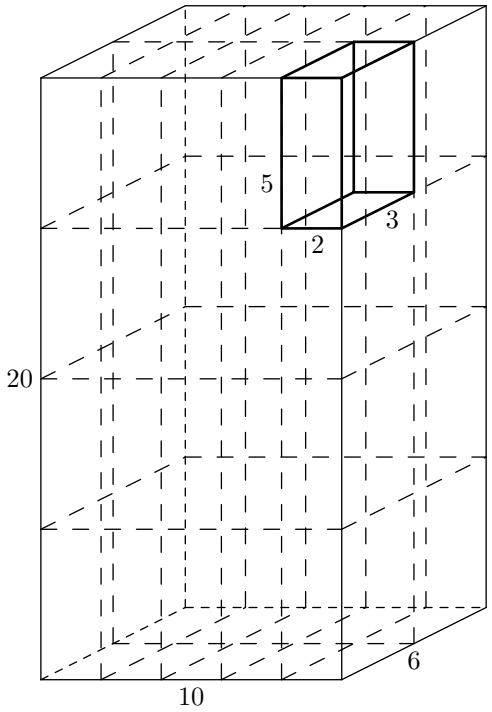
Output three integers w_c, h_c, d_c, the number of cuts to be made along each of the dimensions w, h, and d, respectively, if it is possible to cut the cake, or a single integer -1 otherwise. Note that making zero cuts along a dimension is allowed, too.

Example

standard input	standard output
10 20 6 40	4 3 1

Note

In the first example, the cake will be cut into $5 \cdot 4 \cdot 2 = 40$ pieces of size $2 \times 5 \times 3$ inches.



Problem I. Irrigation Interlock

Time limit:	3 seconds
Memory limit:	1024 megabytes

Two irrigation cooperatives share the same fertile valley. The first cooperative maintains pumps scattered across the fields; the second supervises reservoirs on the surrounding hills. Whenever both cooperatives decide to lay a pair of new pipes, the pipes must intersect — at such an intersection they can install a joint valve. Pipes always follow a straight line segment between a pair of distinct pumps or a pair of distinct reservoirs. Two pipes intersect if they share at least one common point (touching and overlapping pipes are considered intersecting).

You are given the exact coordinates of every pump and every reservoir on a Cartesian plane. For each planning scenario, determine whether the first cooperative can pick two distinct pumps and the second cooperative can pick two distinct reservoirs so that the two straight pipes intersect. If this is possible, report the indices of those pumps and reservoirs; otherwise declare that the project cannot be realized.

Input

The first line contains an integer t ($1 \le t \le 10^5$) — the number of planning scenarios.

For each planning scenario:

The first line contains an integer n ($2 \le n \le 10^5$) — the number of pumps managed by the first cooperative.

Each of the next n lines contains two integers x_i and y_i ($|x_i|, |y_i| \le 10^9$) — the Cartesian coordinates of pump i. The pump locations are distinct.

The next line contains an integer m ($2 \le m \le 10^5$) — the number of reservoirs managed by the second cooperative.

Each of the next m lines contains two integers u_j and v_j ($|u_j|, |v_j| \le 10^9$) — the Cartesian coordinates of reservoir j. The reservoir locations are distinct.

No pump shares its location with any reservoir.

It is guaranteed that the sum of n over all planning scenarios does not exceed $2 \cdot 10^5$ and the sum of m over all planning scenarios does not exceed $2 \cdot 10^5$.

Output

For each planning scenario:

If the first cooperative can choose two pumps and the second cooperative can choose two reservoirs so that the straight pipes connecting each pair intersect, output four integers p_1, p_2, r_1, r_2 — the indices of two chosen pumps ($1 \le p_1, p_2 \le n$; $p_1 \ne p_2$) and two chosen reservoirs ($1 \le r_1, r_2 \le m$; $r_1 \ne r_2$).

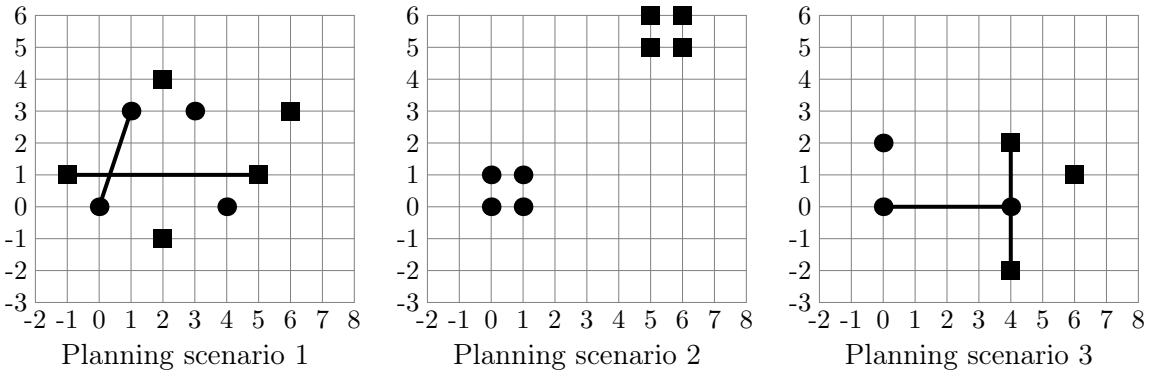
If such an intersection is impossible, output -1.

In case several valid solutions exist, any one of them is acceptable.

Example

standard input	standard output
3 4 0 0 4 0 3 3 1 3 5 -1 1 5 1 2 -1 2 4 6 3 4 0 0 1 0 0 1 1 1 4 5 5 6 5 5 6 6 6 3 0 0 4 0 0 2 3 4 -2 4 2 6 1	1 4 1 2 -1 1 2 1 2

Note



Problem J. Jinx or Jackpot

Time limit: 3 seconds
Memory limit: 1024 megabytes

Jack is in his favourite casino and has 1000 dollars. The casino has literally nothing but a single slot machine. Jack knows the history of this casino. Once upon a time, the future owner of the casino was walking and suddenly saw an array of n integer **choices** $p_1, \ldots, p_n$ each from 0 to 100. He picked an index i ($1 \le i \le n$) uniformly at random and thought that it was a good idea to create a casino in which there is only one slot machine with jackpot probability of $\frac{p_i}{100}$. And he created it.

Jack knows the array of choices $p_1, \ldots, p_n$ that suddenly appeared to the owner during the walk, but he does not know which i the owner picked. However, the chosen index i is fixed forever; the slot machine always uses the same p_i as explained below.

On the slot machine, Jack can bet x dollars, where x is a **non-negative** integer, and pull the lever. Then:

1. With probability $\frac{p_i}{100}$ it will be a jackpot, and the slot machine returns $2x$ dollars to him, so he gains x dollars.
2. With probability $1 - \frac{p_i}{100}$ it will be a jinx, and the slot machine returns nothing to him, so he loses x dollars.

Even if Jack bets 0 dollars, he will understand whether it was a jinx or a jackpot.

Also, the slot machine is not very durable, so Jack can play at most k rounds on it.

Find the maximum expected *profit* Jack can achieve by an optimal strategy. Here a profit is defined as the final amount of money Jack has minus his initial 1000 dollars.

Of course, Jack can't make a bet that is more than his current balance.

Input

The first line contains two integers n and k ($1 \le n \le 100\,000$; $1 \le k \le 30$) — the number of choices and the limit on the number of rounds. The second line contains n integers $p_1, \ldots, p_n$ ($0 \le p_i \le 100$) — the choices.

Output

Output a single real number — the expected profit Jack can achieve by an optimal strategy. Your answer will be considered correct if its absolute or relative error is at most 10^{-4}.

Examples

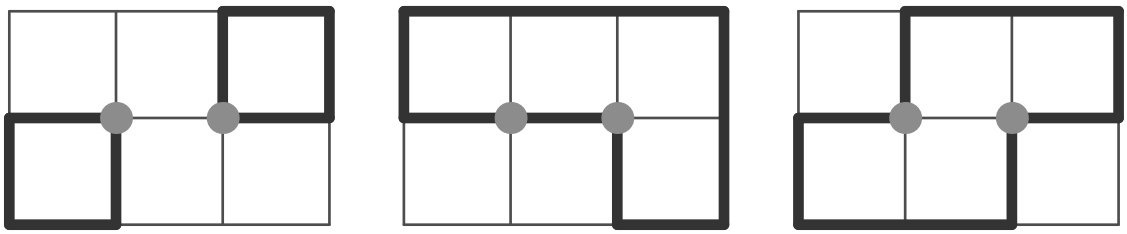
standard input	standard output
2 2 70 30	160
2 30 30 70	12099716.1778528057038784
2 5 40 50	0
6 6 10 20 60 30 40 50	29.40799999999990177457221
1 5 61	1702.708163199999489734182

Problem K. Knit the Grid

Time limit: 3 seconds
Memory limit: 1024 megabytes

The voodoo lady once knitted a magical tapestry. Initially, she took a blank canvas that can be represented as an $r \times c$ grid with r rows and c columns, thus having $(r+1) \times (c+1)$ grid points. Then she did the following operation some number of times: she knitted a cycle on the canvas along the grid lines, passing through each grid point at most once within that cycle. Additionally, no two cycles share any grid point.

In the end, it turned out that exactly one cycle passes through each of the $(r-1) \cdot (c-1)$ inner grid points that don't lie on the canvas' border. Here are some examples of cycle arrangements for $r = 2$, $c = 3$ with the inner grid points highlighted:



Then she left the canvas on the floor overnight. During the night, $r \cdot c$ green frogs hopped on the canvas, with one sitting in each cell. But that was only the beginning of the voodoo lady's troubles! Because then, the old witch came to the canvas, and one-by-one, ripped away every knitted line on the canvas. Every time she ripped away a knitted line segment between two adjacent grid points, the frogs in the cells adjacent to that line segment got startled (there were one or two startled frogs, depending on whether the line segment was on a border or not). When a frog got startled, it instantly changed its color: if the frog was green, it became brown; and if it was brown, it became green again.

If the cycles were arranged as in the pictures above, then the colors would be as follows (greyed out cells represent green frogs and white cells represent brown ones):

When the voodoo lady came back to her canvas, she only saw that there were frogs of two colors on her canvas, but no knitted cycles. From the given arrangement of the frog colors, determine whether it could have been produced by the described process, and if so, help the voodoo lady to restore a possible arrangement of cycles.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers r and c, denoting the dimensions of the grid ($2 \le r, c \le 10^3$).

Each of the next r lines contains a string consisting of c characters `G` or `B` denoting green and brown frogs respectively.

It is guaranteed that the sum of $r \cdot c$ over all test cases does not exceed $2 \cdot 10^6$.

Output

For each test case, on the first line output "YES" if the given frog colors could have been produced by the described process, and "NO" otherwise.

If the answer is YES, output $2r+1$ more lines with binary strings (with 0 and 1 characters). The first $r+1$ of those lines should have length c each and represent the horizontal grid line segments and the next r lines have length $c+1$ each and represent the vertical grid line segments of the answer as explained below.

In the first $r+1$ lines j-th character of the i-th line is 1 if the horizontal grid line segment that is j-th from the left and i-th from the top should have a knitted line along it, and 0 otherwise.

In the next r lines j-th character of the i-th line is 1 if the vertical grid line segment that is j-th from the left and i-th from the top should have a knitted line along it, and 0 otherwise.

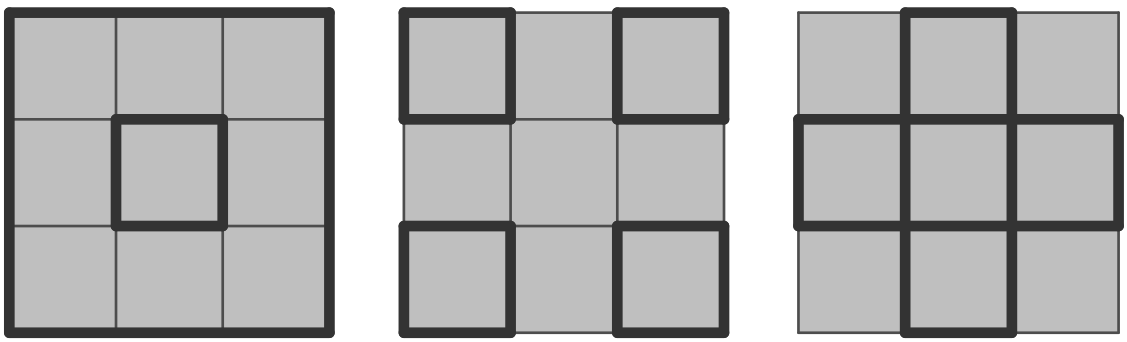
Example

standard input	standard output
3 2 3 BBG GBB 3 3 GGG GGG GGG 3 3 GGG BBB GGG	YES 001 101 100 0011 1100 YES 111 010 010 111 1001 1111 1001 NO

Note

The first test case is the first example of a cycle arrangement from the statement.

In the second sample test case, the output shown in the sample is illustrated in the first picture below. The cycle arrangement in the second picture is also correct, while in the third picture it is not, because some grid points are shared by more than one cycle. Leaving the grid empty would also not be correct, because there would be no cycle passing through inner grid points.



Problem L. LLM Training

Time limit:	3 seconds
Memory limit:	1024 megabytes

You are given a text dataset. Your task is to train LLM (Large Language Model) and find the minimal possible loss. No kidding.

A text dataset is an array of texts $t_1, t_2, \ldots, t_n$. Each text t_i is a sequence of tokens. We define the set of tokens T as the set of all tokens that appear in at least one text t_i. Additionally, for each text t_i, there is a set of positions $L_i \subseteq \{1, 2, \ldots, |t_i|\}$. The token $t_i[j]$ is generated by LLM if $j \in L_i$ and is written by the user if $j \notin L_i$.

Let us define LLM with context size k as a probabilistic model P_k, such that it defines the probability distribution of the next token of the sequence, depending on a context w — a sequence of length between 0 and k (inclusive) whose elements are from T. Thus the probabilistic model P_k is a large table of probabilities $P_k(\text{next}|w)$, defined for any context $w \in T^*$, $0 \le |w| \le k$ and any token $\text{next} \in T$. Conditions $0 \le P_k(\text{next}|w) \le 1$ and $\sum\limits_{\text{next} \in T} P_k(\text{next}|w) = 1$ should be satisfied.

The loss function of LLM with the context size k is the following function defined for P_k:

$$\mathcal{L}_k(P_k) \;=\; \sum_{i=1}^{n} \sum_{j \in L_i} -\log_2 P_k\left(\underbrace{t_i[j]}_{\text{next token}} \;\middle|\; \underbrace{t_i[\max(1, j-k) \,..\, j-1]}_{\text{context}} \right)$$

Here $t_i[l \,..\, r] = t_i[l]t_i[l+1]\ldots t_i[r]$ is the substring from l-th to r-th token, $t_i[1 \,..\, 0]$ is an empty string. So, for each text and for each token that is generated by LLM, we add to the loss the negative logarithm (base 2) of the probability that this token will be generated, depending on the substring of previous k tokens (or the whole prefix, if it has length less than k). If the probability is zero, we assume that the negative logarithm is $+\infty$. This loss function is known as the (base 2) Cross Entropy Loss over the LLM-generated positions. The smaller the loss function value $\mathcal{L}_k(P_k)$, the better LLM P_k is.

For each $0 \le k < \max\limits_{i=1..n} |t_i|$, calculate the minimum possible loss $\mathcal{L}_k(P_k)$ that could be obtained for some P_k — LLM with context size k. It can be proved that this minimum is reachable and is not infinite.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of texts in the dataset. Text descriptions follow.

The first line of the i-th text description contains a single integer m_i ($1 \le m_i \le 3 \cdot 10^5$) — the length of t_i ($m_i = |t_i|$).

The next line contains m_i strings $t_i[1]$, $t_i[2]$, …, $t_i[m_i]$ ($1 \le |t_i[j]| \le 5$) — tokens of the text t_i. Each token consists of symbols with ASCII codes from 33 to 126 (printable characters).

The next line contains a string ℓ_i of m_i letters `U` and `L`, which encodes the set L_i. All positions with the letter `L` are generated by LLM, and all positions with the letter `U` are written by the user. So $L_i = \{j \mid \ell_i[j] = \texttt{L}\}$. It is guaranteed that the last token is generated by LLM, so $\ell_i[m_i] = \texttt{L}$.

It is guaranteed that the sum of m_i for all i ($1 \le i \le n$) does not exceed $3 \cdot 10^5$.

Output

Print $M = \max\limits_{i=1..n} m_i$ real numbers: for each $k = 0, 1, \ldots, M-1$ print the minimum possible loss $\mathcal{L}_k(P_k)$ for all possible P_k — LLM with context size k.

Your answers will be accepted if their absolute or relative errors do not exceed 10^{-6}; formally, if p is your answer, and q is the jury's answer, this should hold: $\frac{|p-q|}{\max\{1,|q|\}} \le 10^{-6}$.

Examples

standard input	standard output
4 5 1 + 1 = 2 UUUUL 5 1 + 2 = 3 UUUUL 5 2 + 1 = 3 UUUUL 5 2 + 2 = 4 UUUUL	6.000000000000 6.000000000000 4.000000000000 4.000000000000 0.000000000000
4 4 N E F <EOS> LLLL 5 N E R C <EOS> LLLLL 6 N E E R C <EOS> LLLLLL 5 I C P C <EOS> LLLLL	55.683674395584 12.490224995673 8.000000000000 8.000000000000 8.000000000000 8.000000000000
1 16 a b a c a b a d b a b d a b a c ULLULLLLLLULLLLL	22.595941331507 12.464393446710 5.245112497837 2.000000000000 0.000000000000 0.000000000000 0.000000000000 0.000000000000 0.000000000000 0.000000000000 0.000000000000 0.000000000000 0.000000000000 0.000000000000 0.000000000000 0.000000000000
2 4 WA WA WA AC LULL 4 AC AC WA AC LLUL	5.509775004327 4.754887502163 4.000000000000 2.000000000000

Problem M. Medical Parity

Time limit: 3 seconds
Memory limit: 1024 megabytes

Nurse Mira works in an allergy clinic. For each patient Mira tests n allergens in a fixed order. The outcome of the tests is written down as a binary string x of length n: for each allergen, 1 means a positive reaction and 0 means no reaction.

To analyze how the reactions are distributed, Mira also writes a *parity control string* for x. For a binary string x of length n, the parity control string y is defined as follows. For every position i ($1 \le i \le n$), let c_i be the number of characters equal to 1 among the first i characters of x (including position i). The parity control string y is the binary string of length n such that $y_i = c_i \mod 2$ for all i ($1 \le i \le n$). In other words, y_i is 1 if c_i is odd and 0 if c_i is even. For example, if $x = 11101$, then $y = 10110$.

Unfortunately, when recording the data, some bits in the test result string and the parity control string may have been written incorrectly. For a given patient, Mira later finds in the system two binary strings x' and y' of the same length n. They were intended to be some true test result string x and its parity control string y, but some bits in x and y might have been flipped during recording. For instance, in the previous example only the 3rd bit in y could have been flipped, resulting in $x' = 11101$ and $y' = 10010$.

In one *bit flip*, a position in one of the two strings is chosen and the bit at this position is flipped (changing 0 to 1 or 1 to 0). Mira wants to know the minimal number of bit flips that could have happened when recording the data.

Formally, you are given two binary strings x' and y' of length n. You want to obtain two strings x and y of length n from x' and y' by flipping some bits in x' and y', so that y is a parity control string of x. Find the minimal possible total number of bit flips needed.

Input

The first line of the input contains the number of test cases t. The $2t$ lines follow — two lines for each test case. The first line of each test case contains a non-empty binary string x' consisting of characters 0 and 1. The second line contains a binary string y' consisting of characters 0 and 1 with the same length as x'.

The total length of all x' strings in the input does not exceed 10^6.

Output

Print t lines — one line for each test case. For each test case, print a single integer — the minimal possible number of bit flips that could have happened when recording the data.

Example

standard input	standard output
3 11101 10110 11101 10010 01100 10110	0 1 2