Problem A. Art of Multiplication

Time limit: 2 seconds
Memory limit: 1024 mebibytes

Let x be an integer represented by n identical digits a in base r. Also, let y be an integer whose representation consists of m digits b in base r.

Find the sum of the digits of the number $z = x \cdot y$ represented in base r, and also output this sum of digits in base r.

Input

The first line contains a single integer t ($1 \le t \le 10\,000$): the number of test cases. The description of the test cases follows.

Each test case consists of a single line containing five integers n, m, a, b, r ($n, m \ge 1$, $n + m \le 10^{18}$, $1 \le a, b < r \le 10$).

Output

For each test case, output a single integer on a separate line: the answer to the problem in base r.

Example

standard input	standard output
5 2 3 6 6 10 6 11 7 7 10 20 25 2 6 10 13 37 1 1 2 57 1 4 2 8	27 84 204 100101 71

Note

In the first test case, we have $x = 66_{10}$, $y = 666_{10}$, $z = 43\,956_{10}$. The sum of the digits of z is 27_{10}.

In the fourth test case, the sum of the digits of z in binary representation is $37_{10} = 100101_2$.

Problem B. Building a Reactor

Time limit:	2 seconds
Memory limit:	1024 mebibytes

Building a reliable and efficient reactor consists of many stages, and you are tasked with helping to design such a reactor.

There is a framework representing a grid of size $n \times m$. Some of its cells contain load-bearing walls for the strength of the structure and cannot be used for any other purposes. The remaining cells can be filled with fuel rods and cooling elements, at most one per cell.

According to safety regulations, each fuel rod must be adjacent to at least one cooling element. Formally, their cells must share a common side.

The more fuel rods there are in the design, the more energy can be extracted from the reactor. Your task is to find any configuration with the maximum number of fuel rods while adhering to all the requirements.

Input

The first line contains two integers n and m ($1 \le n \le 12$, $1 \le m \le 30$): the height and width of the reactor framework.

Each of the next n lines contains m characters describing the reactor cells. The character '`.`' corresponds to a free cell, while the character '`#`' corresponds to a load-bearing wall.

Output

Output nm characters, with m characters in each of the n lines that denote an optimal arrangement of the reactor elements. The characters can take the following values:

- '`#`' if there is a load-bearing wall in the cell;
- '`0`' if you placed a fuel rod in the cell;
- '`1`' if you placed a cooling element in the cell;
- '`.`' if the cell remains empty.

If there are multiple optimal solutions, output any of them.

Example

standard input	standard output
6 9 ....#.... ....#....	001000100 100010001 0010#0100 0010#0100 100010001 001000100

Problem C. Cell Flip Game

Time limit: 1 second
Memory limit: 1024 mebibytes

You are given a binary $n \times m$ matrix A. Basically, it is just a table of n rows and m columns, each cell of which has a value of 0 or 1.

In one move, you can pick a cell (i, j) of this matrix and flip the values in it and in all cells that share a common side with it. Formally, for every cell (s, t) $(1 \le s \le n,\ 1 \le t \le m)$ such that $|s - i| + |t - j| \le 1$, the transformation $A_{s,t} \leftarrow 1 - A_{s,t}$ is applied.

Find a way to make the matrix have only zero elements, or state that it is impossible.

Input

The first line contains two integers n and m $(1 \le n, m \le 500)$, denoting the sizes of the matrix.

The i-th of the following n lines contains a binary string of m digits, corresponding to the i-th row of the matrix.

Output

If there is no solution, print the word `NO` in a single line.

Otherwise, print the word `YES` in the first line. In each of the following n lines, print a binary string of length m. These strings form a matrix with dimensions $n \times m$. Each of its elements has to be equal to 1 if and only if you intend to make a move corresponding to that cell.

Of course, the combination of all picked moves must result in the conversion of the initial matrix into the matrix consisting of only zeros.

If there are several solutions, print any of them.

You may output words in any case, upper or lower.

Example

standard input	standard output
3 4 1000 0001 0100	YES 0100 0110 1100

Problem D. Detecting the Missing Ship

Time limit:	1 second
Memory limit:	1024 mebibytes

This is an interactive problem.

The oceanic expedition of the research ship was going well, but at one point the scientists entered an area where they lost contact with the shore. The first issue is that they cannot transmit the research data back to the mainland. Unfortunately, they will not be able to proceed and will have to return. But the second issue is that due to the loss of communication, there is no reliable way to get back! The group of scientists who remained on the shore has already sent a drone that will find the location of the ship and then guide it back to shore.

The area of the ocean being studied is represented as a table of $n \times n$ cells. The ship occupies $k > 1$ consecutive cells either vertically or horizontally. The rows of the table are numbered from top to bottom from 1 to n, and the columns are numbered from left to right from 1 to n. With one query, the drone can choose a cell (r, c). If the ship has at least one cell in the r-th row **or** in the c-th column, then the search is successfully concluded. Otherwise, after the query, the ship may either stay in place or move one cell in either direction depending on its orientation. The ship cannot leave the studied area of the ocean; all cells of the ship must remain within the table.

An example for $n = 5$, $k = 2$, and checking the drone's position $(r, c) = (1, 2)$ is shown in Figure 1. The arrows indicate where the ship can move after the query.

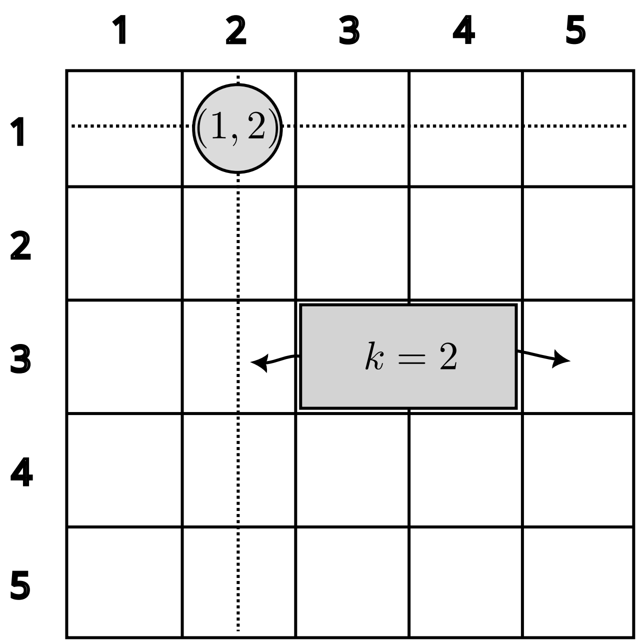

Figure 1. Example of a move

The goal is to find the ship within the minimum possible number of moves by hitting a row or column that definitely contains at least one cell of the ship.

Input

The first line of the input data contains two integers n and k ($3 \le n \le 10$, $2 \le k \le n$), specifying the size of the table and the length of the ship.

Interaction Protocol

To make a query, output two integers r and c ($1 \le r, c \le n$). The response to the query will be written into the standard input stream: you will receive the number 0 if the ship has not yet been found, or the number 1 if it has been found. As soon as you read the number 1, the program must be terminated immediately. If you read the number 0, you may make a new query.

Please note that the interactor is **adaptive** and may not necessarily have a fixed strategy of ship movements.

If you make more queries than the minimum necessary to guarantee finding the ship, or if the query coordinates are outside the studied area of the ocean, your program will be judged as `Wrong Answer`, and the response to the query will be the number -1. Upon reading the response -1, the program must be terminated immediately. If your solution does not terminate immediately after finding the ship, or if it terminates before finding the ship, or if it continues execution after reading -1, it will be considered incorrect, and the verdict may be undefined.

After each query, do not forget to output a newline and flush the buffer to the standard output stream. You can use the following functions:

- `fflush(stdout)` or `std::cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

Example

standard input	standard output
3 2 1	 2 2

Note

In the example, we make one query to cell $(2, 2)$ that allows us to find the ship in one move regardless of its location. In response to this move, the number 1 is read, after which the program terminates because it has found the ship.

Problem E. Encountering a Friend

Time limit:	1 second
Memory limit:	1024 mebibytes

In a local table tennis club, there are n boards (tables), numbered from 1 to n, and $2n$ players.

Today, a tournament of k rounds is played in the club. In each round, exactly n games happen simultaneously: at each board, one game between exactly two players is played. After each round apart from the last one, players change boards as follows:

- The winner of the game at board i moves to board $i+1$, unless $i = n$; in that case, the player stays at the current board.
- The loser of the game at board i moves to board $i-1$, unless $i = 1$; in that case, the player stays at the current board.

Alice and Bob are friends, and both of them participate in the tournament, playing their first games at boards a and b, respectively. Over the course of k games, they may meet and play against each other multiple times.

As they like playing against each other, they asked you: "In this tournament, what can be the maximum possible number of games in which we encounter each other at the same board?" Please help them!

Input

The first line contains one integer t ($1 \le t \le 10^4$), denoting the number of test cases.

Each of the next t lines contains four integers n, k, a, b ($1 \le a, b \le n \le 10^9$, $1 \le k \le 10^9$), denoting that there are n boards and k rounds in the tournament, that Alice starts at board a, and that Bob starts at board b.

Output

For each test case, print a single integer: the maximum number of games that can occur between Alice and Bob over k rounds.

Example

standard input	standard output
3 7 4 2 6 7 5 2 6 4 3 1 4	1 2 1

Problem F. Forgot to Refuel

Time limit: 3 seconds
Memory limit: 1024 mebibytes

Fizz Buzz City is a large settlement, so it's no surprise that it has a ring road to help citizens move among districts. The road is circular and can be traversed both clockwise and counterclockwise. The ring road is exactly ℓ kilometers long and has ℓ intersections on it. Intersections are placed such that the distance along the road between any two neighboring intersections is exactly 1 kilometer.

One day, Fiodar was driving the ring road while remembering he needed to refuel his car. Unfortunately, he missed the turn to a fuel station! Of course, he couldn't just take a U-turn and go back due to traffic regulations. So he stopped immediately after the missed turn and thought: "Alright, I'll drive along the ring road in the same direction straight to the next fuel station. But what if it's too far?"

Fiodar was new to Fizz Buzz City, so he didn't know the exact placement of fuel stations. But he knew there were n fuel stations on the ring road in total. He also knew that fuel stations were located only at intersections, and no two fuel stations could be located at the same intersection.

"I wonder, what is the largest distance I have to drive *for sure* to meet at least one fuel station? Considering that I don't know how fuel stations are placed, what is the largest distance in the average case if any placement is equally probable?" Fiodar thought.

Could you help him answer this question? Find the expected value of the maximum distance he has to cover to meet at least one fuel station, no matter where he is currently located and which direction he is heading. Consider all valid placements of fuel stations equally probable.

Input

The only line contains two integers ℓ and n ($2 \le n \le \ell \le 10^6$): the length of the ring road in kilometers and the number of fuel stations.

Output

Print a single integer: the desired expected value of the maximum distance modulo 998 244 353.

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

Examples

standard input	standard output
2 2	1
5 3	499122179
20 5	796329095

Note

In the first example, no matter how fuel stations are placed, the maximum distance from any point on the road to a fuel station is 1.

In the second example, there are 5 placements with distances $(1, 2, 2)$ (maximum is 2) and 5 placements with distances $(1, 1, 3)$ (maximum is 3). Therefore, the fractional answer is $\frac{5 \cdot 2 + 5 \cdot 3}{5+5} = \frac{5}{2}$.

Problem G. Guaranteed Medal

Time limit: 1 second
Memory limit: 1024 mebibytes

As you might know, team programming competitions usually follow the same set of rules: there are t participating teams of 3 people each; p problems, denoted by the first p uppercase English letters; and 5 hours to solve them using 1 computer per team. To determine the final standings, teams are compared using the following criteria, in the order of importance:

- A team with more solved problems is placed higher.
- A team with a smaller *penalty time* is placed higher. The penalty time is defined as the sum of the following values over all problems solved by the team: the submission time from the start of the contest in minutes for the first successful solution, plus 20 minutes for every incorrect solution before that successful submission.
- A team with an earlier *decisive submission* is placed higher. A submission is *decisive* if it achieved the n-th solved problem for the team, where the team has solved exactly n problems in total.

Recently, a big championship concluded, with medals awarded to the top 12 teams. A team T is said to have *guaranteed* its medal with the submission S if, after removing all of T's submissions that occurred after S, T still receives a medal, but removing S itself in addition causes T to no longer receive a medal.

Given the log of submissions, find, for each of the awarded teams, when they guaranteed their medals.

Input

The first line contains one integer T ($1 \le T$), denoting the number of test cases that follow.

The first line of a test case description contains two integers p and s ($1 \le p \le 26$, $13 \le s \le 10^5$): the number of problems and the number of submissions. Each of the next s lines describes a submission in the format *team problem time verdict*, in the order they were received by the testing system.

- *team* is a string of 1 to 50 characters, which can be lowercase or uppercase English letters, digits, or the characters '`-`', '`_`', '`.`' (the minus sign, underscore, dot). It denotes the name of the team that made the submission.

  No two teams have the same name, and every participating team made at least one submission.

- *problem* is a single uppercase English letter. If it is the i-th letter in the alphabetical order, it indicates the submission was for the i-th problem of the contest. It is guaranteed that $1 \le i \le p$.
- *time* is a 4-character string in the format h:mm, where all characters except one are digits, denoting the elapsed time from the start of the contest: h hours and mm minutes ($0 \le h \le 5$, $00 \le mm \le 59$, $h \cdot 60 + mm \le 300$).

  It is guaranteed that, within a single test case, *time* for each submission is not less than *time* for the previous one. Note, however, that the equality of *time* doesn't imply that the submissions occurred simultaneously: the one listed earlier happened earlier.

- *verdict* is a string of two uppercase English letters. The submission was correct if and only if the verdict is `OK`. For the purposes of this problem, there are no verdicts that are ignored when calculating the penalty time (for example, `CE` also counts as an incorrect attempt).

It is guaranteed that at least 13 teams solved at least one problem, the sum of s across a single file does not exceed 10^5, and the sum of all $|team|$ across a single file does not exceed 10^6.

Output

For each team that received a medal, print a single line with two strings: the team name and the time of the submission that guaranteed its medal, in the format h:mm. List the teams in the same order as their guaranteeing submissions appear in the input.

Example

standard input	standard output

<pre>1
3 34
Lucky_I_of_Tech C 0:01 WA
Abbr._U_of_Tech A 0:15 OK
Bioinformatics_U A 0:20 OK
M._University A 0:30 OK
UN_de_Ingenieria A 0:35 OK
College_No_1234 A 0:40 OK
Discrete_Math_U A 0:45 OK
Ecole_Perpendiculaire A 0:50 OK
FFT_University A 1:00 OK
Great_Staff_U A 1:11 OK
Hard_Problems_U A 1:20 OK
Lucky_I_of_Tech B 1:30 OK
Never_Medal_U A 1:30 OK
Lucky_I_of_Tech B 1:30 OK
Unfreezing_U A 1:40 OK
Lucky_I_of_Tech B 1:40 TL
Univ._of_Toyoko A 2:11 OK
Univ._of_Toyoko B 2:50 RE
Univ._of_Toyoko B 2:50 ML
Univ._of_Toyoko B 2:51 WA
Univ._of_Toyoko B 2:51 OK
Bioinformatics_U B 3:49 OK
Abbr._U_of_Tech B 3:50 OK
M._University B 3:59 OK
UN_de_Ingenieria B 4:00 OK
College_No_1234 B 4:10 OK
Discrete_Math_U B 4:12 OK
Ecole_Perpendiculaire B 4:14 OK
FFT_University B 4:16 OK
Great_Staff_U B 4:18 OK
Hard_Problems_U B 4:25 OK
Lucky_I_of_Tech A 4:32 OK
Never_Medal_U B 4:32 OK
Unfreezing_U B 4:59 OK</pre>
	<pre>Univ._of_Toyoko 2:51
	Bioinformatics_U 3:49
	Abbr._U_of_Tech 3:50
	M._University 3:59
	UN_de_Ingenieria 4:00
	College_No_1234 4:10
	Discrete_Math_U 4:12
	Ecole_Perpendiculaire 4:14
	FFT_University 4:16
	Great_Staff_U 4:18
	Hard_Problems_U 4:25
	Lucky_I_of_Tech 4:32</pre>

Note

In the example, `Lucky_I_of_Tech`, `Univ._of_Toyoko`, and `Never_Medal_U` are tied with 2 solved problems and 362 minutes of penalty time. However, `Never_Medal_U` submitted their second successful solution later than the two other teams, so they did not receive a medal.

Note that, for the purposes of this problem, the competition rules may differ slightly from those used in the actual ICPC World Finals or any of its regional competitions.

Problem H. Helpful Bits

Time limit: 2 seconds
Memory limit: 1024 mebibytes

Imagine that you are asked to find the length of the shortest path between several pairs of vertices in a weighted undirected graph.

Sounds too easy, doesn't it? We think so, too.

That's why in this problem, your program will be run twice on each test. During the first run, you can read the weights of the edges. During the second run, you will be only given the graph without weights of its edges, and you will be asked to find the weight of the shortest path between several pairs of vertices.

"How would I know the shortest path without weights?" you might ask. That's a perfectly reasonable question, so we'll make it easier for you:

- If you are given m weights during the first run, you can output at most $12m$ bits as a binary string, and they will be passed to the second run of your program as is.
- The shortest path lengths don't have to be precise, but can differ from the actual value X by at most 0.1% of it. In other words, your answer Y has to satisfy the condition $0.999X \le Y \le 1.001X$.

Input

Your program will be run **twice** on each test.

The first line contains one integer T ($1 \le |T| \le 10^5$). Its absolute value denotes the number of test cases. If $T < 0$, it's the first run; otherwise, it's the second run. Then, $|T|$ test cases follow in the following format.

If this is the first run:

The next line contains one integer m ($1 \le m \le 10^5$), denoting the number of edges in the graph.

The next line contains m integers $w_1, w_2, \ldots, w_m$ ($1 \le w_i \le 10^6$), denoting the weights of the edges.

If this is the second run:

The next line contains four integers n, m, q, b ($2 \le n \le 10^5$, $1 \le m \le 10^5$, $1 \le q \le 10^5$, $0 \le b \le 12m$), denoting the number of vertices, the number of edges in the graph, the number of shortest path length queries, and the number of bits you sent during the first run, respectively.

The next line contains a binary string of length b: the bits you printed during the first run.

The i-th of the next m lines contains two integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$): the endpoints of the next edge. The weight of this edge is equal to w_i from the first run (but is not given during the second run).

Each of the next q lines contains two integers s, f ($1 \le s, f \le n$), which means that you are asked to find the length of the shortest path from vertex s to vertex f.

It is guaranteed that the value of m given during both runs is the same, that no pair of vertices is directly connected more than once, and that both the sum of n and the sum of $m \cdot q$ over all test cases in a single file do not exceed $3 \cdot 10^5$.

Output

For each test case:

If this is the first run:

Print one integer b ($0 \le b \le 12m$): the number of bits you intend to pass to the second run.

If $b > 0$, print a single binary string of length b in the next line: the bits themselves. They will be passed intact to the second run.

If this is the second run:

For each of the q queries, print the length of the shortest path between two requested vertices, or -1 if such a path does not exist. Your answer doesn't have to be an integer and can be a fractional number with at most 15 digits after the decimal separator. Your answer will be accepted if its **absolute** difference from the correct answer X is at most $X \cdot 10^{-3}$.

Example

standard input	standard output
-2 3 4 5 6 1 1	36 000000000100000000000101000000000110 12 000000000001
- - - - - - - - - - - - - - - - - - -	- - - - - - - - - - - - - - - - - - -
2 4 3 5 36 000000000100000000000101000000000110 1 2 2 3 3 1 1 1 1 2 3 1 2 3 2 4 2 1 1 12 000000000001 1 2 1 2	0.00000 4.00000 6.00000 5.00000 -1.00000 1.00000

Note

In the example, the input and output data are listed for both runs of the program, separated with a dashed line and additional empty lines. Neither dashed lines nor empty lines are present in the actual input and output data.

For both test cases of the example, during the first run, the solution decided to use all available bits and write all the weights as 12-digit binary numbers. Then, during the second run, the solution worked out what the weights of all the edges were and printed all the requested shortest path lengths, including the absence of a path between vertices 2 and 4.

Note that when you view examples in the testing system, input data might not correspond to the exact input data for any of the runs, as those are generated from what is shown in the internal format of tests.

Problem I. Ivan Tsarevich in the Magical Grove

Time limit: 3 seconds
Memory limit: 1024 mebibytes

Ivan Tsarevich, in search of the death of Koschei the Deathless, has come to a magical grove. According to Baba Yaga, there are n wise oaks in it, among which exactly k oaks always tell the truth, while the others always lie.

To find out where to go next, the Tsarevich needs to learn the further route from the oaks. However, he must be extremely cautious, as if he follows the advice of one of the lying oaks, he will wander into a swamp where the one-eyed Likho would ambush him.

To figure out which oaks he can trust, Ivan has asked each oak a question in the form: "Oak number i, tell me, does the oak number a_i always tell the truth?" However, he cannot figure out what to do with the answers he has received.

Help Ivan Tsarevich draw conclusions from the questions. For each oak, determine whether it always tells the truth, always lies, or if the information is insufficient. If Baba Yaga has deceived Ivan Tsarevich and there is no possible configuration of oaks that satisfies the input data, output a single number -2.

Input

The first line contains two integers n and k ($1 \le n \le 5 \cdot 10^5$, $0 \le k \le n$), denoting the number of oaks and how many of them always tell the truth, respectively.

The i-th of the next n lines contains two integers a_i and b_i ($1 \le a_i \le n$, $0 \le b_i \le 1$), denoting that the oak number i has been asked whether the oak number a_i always tells the truth, and the answer has been "yes" if $b_i = 1$, or "no" otherwise.

Output

If the input data are contradictory, print a single integer -2.

Otherwise, print n integers $t_1, t_2, \ldots, t_n$: conclusions about whether the oaks tell the truth. They can take the following values:

- $t_i = 1$ if the oak number i always tells the truth;
- $t_i = -1$ if the oak number i always lies;
- $t_i = 0$ if it is impossible to determine whether the oak number i always lies or tells the truth from the available data.

Examples

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

Problem J. Jolly Wheel

Time limit: 2 seconds
Memory limit: 1024 mebibytes

You have a round wheel. The length of its perimeter along the outer circumference is k meters.

You drew a long straight line on a flat surface and placed the wheel on it, then performed the following action n times: roll the wheel along the line, alternating the direction. That is, if during the i-th action the wheel rolled forwards, then during the $(i+1)$-th action it rolled backwards, and vice versa. During the i-th action, the wheel traveled exactly a_i meters.

Now you would like to know the minimum and maximum number of times any point on the outer circumference of the wheel touched the surface.

Input

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

Then, t test case descriptions follow. The first line of a description contains two integers n and k ($1 \le n \le 3 \cdot 10^5$, $1 \le k \le 10^{18}$), denoting the number of actions and the perimeter of the wheel, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{12}$), denoting the distances the wheel traveled during the first, second, …, n-th actions.

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

Output

For each test case, print two integers: the minimum and maximum number of times any point on the outer circumference of the wheel touched the surface.

Example

standard input	standard output
2 3 4 4 8 4 2 7 6 5	4 5 0 2

Note

Note that in this problem, the input and output data can be quite large. It is recommended to use means of speeding up operations with input and output data that are available in your programming language.

Problem K. Kitchen

Time limit: 2 seconds
Memory limit: 1024 mebibytes

Upon entering his kitchen, Petya discovered a terrible surprise: it was infested with cockroaches!

Without wasting a moment, he placed poisonous baits in several spots to exterminate them. Since he did this in a hurry, there was a chance that some cockroaches would survive. Help him predict the outcome.

By the rectangular shape of their bodies, Petya was able to determine that the cockroaches belonged to a rare species of Manhattan cockroaches. If such a cockroach runs from point (x_1, y_1) to point (x_2, y_2), it first runs in a straight line to point (x_2, y_1), and then it runs from there in a straight line to (x_2, y_2).

Each cockroach chooses the bait that minimizes its travel time (and if there are multiple such baits, it chooses the one with the lexicographically smallest coordinates (x_j, y_j)) and moves towards it at a speed of 1 meter per second. When a cockroach reaches the bait, it instantly eats it completely, after which all cockroaches that were running towards it change their target according to the algorithm described above. If several cockroaches reach the same bait simultaneously, the one whose original coordinates were given in the input data first eats it in the heat of intra-species competition.

Since these cockroaches are extraordinarily resilient, they can survive if they eat fewer than a pieces of poison, and even after eating that amount or more, they remain active for t seconds before dying. If a cockroach reaches the bait exactly at the moment of its death, it still has time to eat it.

Input

The first line contains four integers n, m, a, t ($1 \le n \le 100$, $1 \le m \le 1000$, $1 \le a \le 1000$, $0 \le t \le 10^9$), meaning that there are n cockroaches and m baits; after eating a baits, a cockroach will die in t seconds.

The second line contains two integers W and H ($1 \le W, H \le 10^9$): the dimensions of Petya's rectangular kitchen in meters.

Each of the following n lines contains two integers x, y ($0 \le x \le W$, $0 \le y \le H$): the initial coordinates of another cockroach.

Each of the following m lines contains two integers x, y ($0 \le x \le W$, $0 \le y \le H$): the coordinates of another bait. It is guaranteed that no two baits are located at the same point.

Output

For each cockroach, output a single integer in a separate line: the number of seconds after which it will die from the poison, or `-1` if the cockroach survives. The answers for the cockroaches should be given in the same order as the coordinates of the cockroaches appear in the input data.

Example

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

Note

The coordinates (x_1, y_1) are *lexicographically smaller* than the coordinates (x_2, y_2) if either $x_1 < x_2$, or both $x_1 = x_2$ and $y_1 < y_2$.

Problem L. Layered Caesar Salad

Time limit: 1 second
Memory limit: 1024 mebibytes

"What solution did you have?"
"Brute."
"Me too, brute."
"*Et tu, brute?*"

From a contestants' discussion

Behold an innovative encryption method, inspired by the balance of ancient wisdom and modern simplicity!

You must be familiar with the Caesar cipher. It is exceptionally simple: each letter of the message is shifted right in the alphabet by the same number of positions. If shifting goes beyond the alphabet, it proceeds from its beginning. For example, the word `fusion` with a shift of 6 becomes the word `layout`. To decrypt the message, each letter must be shifted left by the same number.

The new encryption method is called the *Caesar Salad cipher*. It is very similar to its ancient prototype but is more reliable! It is applied to words consisting of letters from `a` to `z`. Each letter of the Latin alphabet corresponds to a numerical code from 0 to 25: the letter `a` has code 0, the letter `b` has code 1, `c` has code 2, and so on up to the letter `z` that has code 25. The Caesar Salad cipher works as follows:

1. Let $(a_1, a_2, \ldots, a_k)$ be the numerical codes of the letters of the original message of length k. For example, the word `delta` corresponds to the sequence of codes $(3, 4, 11, 19, 0)$.
2. Choose a shift x, an integer from 1 to 25 inclusive. Note that, just as in the original Caesar cipher, a zero shift cannot be chosen for security reasons.
3. Assign $b_1 := (a_1 + x) \mod 26$.
4. Next, for each i from 2 to k, assign $b_i := (a_i + b_{i-1} + x) \mod 26$.
5. The sequence of codes $(b_1, b_2, \ldots, b_k)$ corresponds to the encrypted message. So, after encrypting the word `delta` with a shift of $x = 20$, we get $(23, 21, 0, 13, 7)$ which corresponds to `xvanh`.

And now, without further ado, it's time to apply your new knowledge in practice! You are to develop a protocol for transmitting a digital key. A digital key consists of n words, each containing k Latin letters. Among these words, there may be duplicates, but in general, their order does not matter. Before transmission over open channels, n new words of length k are mixed in, which are the original words passed through the Caesar Salad cipher. Each of the n words is allowed to be encrypted with different shifts. The resulting multiset of $2n$ words, shuffled in arbitrary order, is called the digital key *with admixture*.

Your task is to implement both parts of the protocol. In the first part, based on the given digital key, you need to generate a key with admixture, and in the second part, extract all the words of the original digital key from the key with admixture generated by your algorithm.

Input

Your program will be run **twice** on each test. In the first run, the input will consist of the original data, and in the second run, there will be the data obtained after your encryption.

Each test contains multiple test cases. The first line contains one word S of uppercase Latin letters and one integer t ($1 \le t \le 1000$): the number of test cases. If $S =$ `ENCODE`, this is the **first run**, and for each test case, you are required to generate a digital key with admixture. If $S =$ `DECODE`, this is the **second run**, and for each test case, you are required to extract the words of the original digital key from the key with admixture. The description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le n \le 25$, $1 \le k \le 20$): the number of words in the original digital key and the length of each of these words.

If $S =$ ENCODE, each of the next n lines contains one word of k lowercase Latin letters. These n words define the original digital key. Some words may be identical.

If $S =$ DECODE, each of the next $2n$ lines contains one word of k lowercase Latin letters. These $2n$ words define the digital key with admixture, output by your program in the first run. The order of words in the key with admixture may be arbitrary.

Output

If $S =$ ENCODE, for each test case, output n words of length k: the words encrypted with the Caesar Salad cipher. Different shifts may be chosen for each word, but the output order of the encrypted words must correspond to the order of the original words. These encrypted words will be mixed with the n original words before the second run.

If $S =$ DECODE, for each test case, output n words of length k: the restored digital key that must match the original key. You may output the words of the digital key in any order during the second run.

The case of letters matters: the output Latin letters must be lowercase.

Example

standard input	standard output
ENCODE 2 4 5 delta alpha alpha prime 1 20 petrozavodskprogcamp	xvanh zjxdc kfevf rkuio vfebvaghbkiytqkwekcx
DECODE 2 4 5 alpha prime kfevf delta zjxdc alpha rkuio xvanh 1 20 vfebvaghbkiytqkwekcx petrozavodskprogcamp	alpha delta prime alpha petrozavodskprogcamp

Note

The example illustrates the format of input and output data for both runs. The descriptions of data for the first and second runs are separated by a blank line for clarity.

For the first test case, shifts of 20, 25, 10, and 2 were chosen for the words, respectively.

For the second test case, after the first run, the word was encrypted with a shift of 6.

Problem M. May We Answer Your Questions Right?

Time limit: 4 seconds
Memory limit: 1024 mebibytes

Working in the *Right Place to Ask* city information desk is no easy task! As an employee, you constantly receive questions that are hardly answerable instantly.

Vasya also doesn't like answering questions. However, he likes it when computer programs do. Right now, he has an array a of n integers $a_1, a_2, \ldots, a_n$, and he wants to write a program that can respond to the following queries:

- `1` i val: assign the value val to the element a_i;
- `2` l r: calculate *Vasya's sum* for the numbers $a_l, a_{l+1}, \ldots, a_r$.

Vasya does not consider all numbers equally good. He prefers numbers that are to the right of other numbers, and exactly twice as much! Therefore, he defines *Vasya's sum* for the numbers $x_1, x_2, \ldots, x_k$ as the value

$$\sum_{i=1}^{k} 2^{i-1} \cdot x_i = x_1 + 2x_2 + 4x_3 + \ldots + 2^{k-1}x_k.$$

Even if Vasya cannot write such a program, it's no problem: where, if not in the *Right Place to Ask*, can they help you calculate something in favor of the numbers on the right? So, now the burden of solving this has fallen on you! To facilitate the calculations, you only need to determine whether each *Vasya's sum* is positive, negative, or zero.

Input

The first line of input contains a single integer t ($1 \le t \le 5 \cdot 10^5$): the number of test cases. The following t descriptions of test cases are in the format described below.

The first line of a test case contains two integers n and q ($1 \le n, q \le 5 \cdot 10^5$): the number of integers in Vasya's array and the number of queries, respectively.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$): the elements of Vasya's array in left-to-right order.

Each of the following q lines describes a query either in the format `1` i val or in the format `2` L R ($1 \le i \le n$, $-10^9 \le val \le 10^9$, $1 \le L \le R \le n$). A query of the first type means that the value of the element a_i must be changed to val. A query of the second type means that you need to find the sign of *Vasya's sum* for the numbers $a_L, a_{L+1}, \ldots, a_R$ (exactly in that order, of course). All numbers in the queries are integers.

It is guaranteed that the sums of each of the values n and q across all test cases do not exceed $5 \cdot 10^5$, and that there is at least one query of the second type in each test case.

Output

For each test case, output one integer for each query of type `2`. If *Vasya's sum* in the given query is positive, output `1`; if negative, output `-1`; otherwise, output `0`.

Example

standard input	standard output
2 4 7 9 -4 2 -1 2 1 4 2 2 3 2 2 4 1 1 8 2 1 4 1 2 -5 2 1 4 7 1 2026 2 5 -14 59 59 -75 2 1 7	1 0 -1 0 -1 -1

Note

In the first test case, *Vasya's sums* are 1, 0, -4, 0, -2, respectively.

In the second test case, *Vasya's sum* is -30.

Note that in this problem, the input and output data can be quite large. It is recommended to use means of speeding up operations with input and output data that are available in your programming language.