Problem A. Surrender to My Will

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

League of Legends (LOL) is a famous MOBA game where, in the standard mode, two teams of 5 players each control their own heroes and engage in battles with the objective of destroying the enemy base.

However, many battles can reach a point where it becomes apparent halfway through that one team is on the brink of defeat. Therefore, LOL has a surrender mechanism.

Specifically, when a member of a team initiates a surrender vote, each member of the team can vote either for or against it. If the number of votes in favor is 4 or more, the surrender is considered successful; otherwise, it fails.

Here is an example of a successful surrender.

However, often the outcome of the surrender vote can be determined before all team members have voted. You are given the current state of votes, represented as a string S of length 5, where each character can be `Y` (for a vote in favor), `N` (for a vote against), or `-` (indicating no vote yet). The string S satisfies the following constraints:

- The first character is `Y`, indicating the initiator of the surrender vote has voted in favor.
- There exists an index $1 \le L \le 5$ such that for all indices $1 \le j \le i$, $S[j] \in \{\texttt{Y}, \texttt{N}\}$, and for all indices $j > i$, $S[j] = \texttt{-}$.

Your task is to determine if the final outcome of the surrender vote can be determined based on the given state S. If it can be determined, output:

- 1 if the surrender is successful,
- -1 if the surrender fails.

If it cannot be determined based on the given state S, output 0.

Input

A string S of length 5 consisting only of characters `Y`, `N`, and `-`.

Output

If the result is determined, print 1 if the vote succeeds and -1 otherwise. If the result is not determined and is based on the votes of some players, print 0.

Examples

standard input	standard output
YYYY-	1
YNNY-	-1
Y----	0

Note

In sample 1, the surrender is guaranteed to succeed with four votes in favor, regardless of the fifth player's vote.

In sample 2, the surrender is guaranteed to fail with two votes against, regardless of the fifth player's vote.

In sample 3, the outcome is uncertain. For example, the surrender could succeed if any three of the last four players vote in favor, and it could fail if any two of the last four players vote against.

Problem B. pair<pair<int,int>,int>

Input file:	standard input
Output file:	standard output
Time limit:	3 seconds
Memory limit:	1024 megabytes

Perhaps you are familiar with `std::pair` in C++: it creates a pair of two types and provides basic functionalities.

For instance, for the types `double` and `int`, a `pair` of them would be `pair<double,int>`. You can access the `double` member by `.first` and the `int` member by `.second`. Similarly, you can access the `int` member of a `pair<int,double>` by `.first`, and the `double` member by `.second`.

Using a `pair` is convenient when you need a quick combination of two data types. However, it can become quite complex when using nested pairs, such as `pair` of a `pair`, `pair` of a `pair` of a `pair`, etc. Your task is to write a helper program that displays the types of members in these nested pairs.

For this problem, we only consider types created recursively by the following rules:

- Basic types are `double` and `int`.
- For two possible types A and B, `pair<`A`,`B`>` is a possible type.

Input

The first line contains two integers n, q ($1 \le n, q \le 1000$), the number of variable declarations and queries.

Each of the following n lines contains a line in the following format:

`[type name] [variable name];`

where the `type name` is a string denoted a type defined above, and the `variable name` is a non-empty string consisting of letters (uppercase or lowercase, **case sensitive**), digits, and underscores (`_`, ascii 95), which must not start with a digit. There will not be any extra spaces except one single space between `type name` and `variable name`.

For the next q lines, each contains a query string. Possible formats are defined recursively by the following rules:

- `variable name` is a possible query.
- If A is a possible query and A is a pair, A`.first` and A`.second` are possible queries.

It is guaranteed that each line of input contains at most 5000 characters, no two variables share the same name, and each query must correspond correctly to a previously defined variable.

Output

For each query, print a line of string in the same format of `type name` in the input, which must not contain extra spaces.

Example

standard input	standard output
3 3 int a; pair<int,int> E6; pair<pair<double,int>,double> __Fukami; a E6 __Fukami.first.second	int pair<int,int> int

Problem C. Capability Expectation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

In the world where competitive programming competitions are with no limit on the number of participants, n participants have decided to form a team for the Consistent and Continuous Pigeon Competition tomorrow. As the name suggests, each of the participants might independently decide not to show up on the day of the contest, with probability p_i.

Now let's talk about the competition. Since it's not an algorithmic competition but a programming one, there's nothing about algorithmic ability. Instead, each participant is associated with their problem-solving and reading abilities, represented by x_i and y_i, respectively.

The team can only solve the problems within its *capability*. The team's *capability* to solve problems is geometrically represented by an area of points of the weighted average of participants: if there are k participants (renumbered to $1 \dots k$) in the team, the *capability* is

$$\text{capability} = \left\{ (\mathbf{w} \cdot \mathbf{x}, \mathbf{w} \cdot \mathbf{y}) \mid \mathbf{w} \in \mathbb{R}^k_{\geq 0}, \sum_{i=1}^{k} w_i = 1 \right\},$$

where $\cdot$ is the dot product: $\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{k} a_i b_i$.

Given all information about the participants, please calculate the expected area of *capability* of the team formed by the participants who do show up.

Input

The first line of the input contains an integer T ($1 \leq T \leq 100$) , the number of test cases.

For each test case, the first line contains an integer n ($1 \leq n \leq 2000$) , the number of potential participants. Each of the next n lines for a case contains three values: a real number p_i ($0 \leq p_i \leq 1$) with at most three decimal places, and two integers x_i, y_i ($0 \leq x_i, y_i \leq 10^9$) .

It's guaranteed that in any test case, no two points are equal and no three points are collinear. Also, the sum of all n across the cases does not exceed 2000.

Output

Output a single decimal real number, representing the expected area of *capability* of the team formed by the participants who do show up.

Your answer will be accepted if it is within an absolute or relative error of 10^{-6}.

Example

standard input	standard output
5 3 0 0 0 0 0 1 0 1 0 3 0.5 0 0 0.5 0 1 0.5 1 0 3 1 0 0 1 0 1 1 1 0 4 0.5 0 0 0.5 0 1 0.5 1 0 0.5 1 1 3 0.114 5 14 0.191 9 810 0.192 60 817	0.5000000000 0.0625000000 0.0000000000 0.1875000000 11747.5474033280

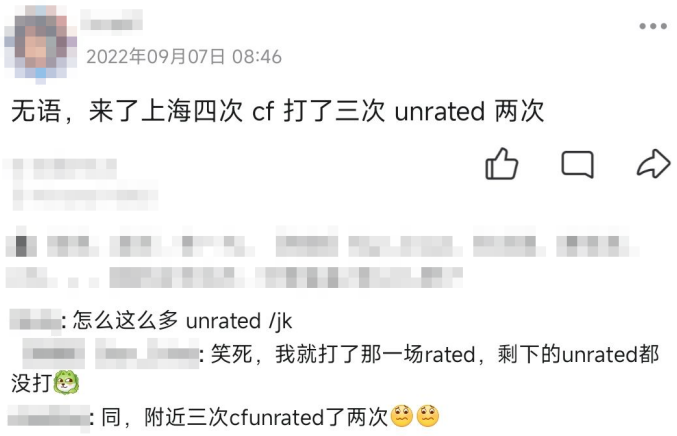
Note

For the second test case, it's only possible to have positive area if all participants do show up. The area is 0.5, and the probability is $0.5 \times 0.5 \times 0.5$. Therefore, the expected area is $0.5 \times 0.5 \times 0.5 \times 0.5 = 0.0625$.

Problem D. Is it rated?

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Isn't it frustrating when a contest round where you've excelled is declared unrated? This may happen due to incorrect problems, long queues, server issues, or simply because some participants disliked the round?!



Post: "I'm speechless. (Since last month) there have been four rounds; I've participated in three, and two were unrated."

You suspect a conspiracy: a ruthless, dark hand manipulates the rating system to favor certain individuals. Imagine you are this manipulator, but also a contestant on the Qniversal Cup contest platform. You intend to use the power of making rounds unrated to boost your ratings. There are n contests, and your performance in each contest is denoted by p_i. You can issue up to m apology announcements, each making a round unrated while all the other contests remain rated.

The final rating is calculated as follows:

- Start with the initial rating r_0:
$$r \leftarrow r_0.$$
- For each rated contest i in increasing order, update the rating as a weighted average of the old rating and the performance:
$$r \leftarrow k \cdot p_i + (1 - k) \cdot r,$$
where k is a predetermined update coefficient.

Your goal is to maximize your final rating. Determine the highest possible rating you can achieve after making at most m rounds unrated.

Although the ratings displayed on contest platforms are usually integers, you know that internally, ratings are calculated using real numbers. Therefore, you need to report the real rating, accurate up to an absolute or relative error of 10^{-9}.

Input

The first line contains a single integer T ($1 \le T \le 1000$), denoting the number of test cases.

For each test case, there are three lines:

- The first line contains two integers n, m ($1 \le m \le n \le 10^5$) and a real number k ($0.1 \le k \le 1$), where n is the number of contests, m is the maximum number of apology announcements you can make, and k is the update coefficient with up to 3 digits after the decimal point.

- The second line contains a single integer r_0 ($0 \le r_0 \le 10^5$), denoting the initial rating.
- The third line contains n integers, $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le 10^5$), representing your performance in each contest.

It is guaranteed that the sum of n among all T cases will not exceed 3×10^5.

Output

For each test case, output a single real number representing the highest possible rating you can achieve, which should be printed in decimal form and have at most 20 digits after the decimal point.

Your answer will be accepted if it is within an absolute or relative error of 10^{-9}.

Example

standard input	standard output
3 2 1 0.5 2400 2700 2100 3 2 0.123 1500 1555 1666 1777 3 3 0.456 3000 0 1 2	2550.000000000000 1557.180743685000 3000.000000000000

Problem E. Cherry on Top

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Recently, you may have heard of the game *Honkai: Star Rail*. In short, it is a turn-based game where the player's characters and enemies take turns to act in battle.

For simplicity, let's focus on a simplified scenario as follows: there is only one enemy with a specific initial HP h, and the goal is to defeat the enemy by reducing its HP to zero or below. In each turn, the enemy will do nothing while the player's character, *Qingque*, can act once.

Qingque is a unique character whose core mechanism involves *Celestial Jade*. There are three types of jades in *Celestial Jade*, numbered 1, 2, and 3. *Qingque* can hold up to four jades at once, and her damage dealt to the enemy depends on the jades in her hand. At the start of the battle, *Qingque* has no jades.

During each turn, *Qingque* will do the following in order:

1. Draw two jades, or only one if she already has three jades in hand.
2. Attack the enemy. If she has four jades of the same type, she will discard them all and deal a damage to the enemy. Otherwise, she will discard one jade of the type with the lowest count in her hand and deal b damage to the enemy. If there are multiple types with the same minimal count, she will discard the type with the smallest number.

`AntiLeaf` has discovered that the jades *Qingque* draws in a battle follow a predetermined pattern. The game server keeps a sequence of jades of length n, denoted as $(s_1,\ s_2,\ \ldots,\ s_n)$. For each battle, the server will choose a continuous subsequence of s as the hidden draw pile. Precisely, if the chosen subsequence is $(s_l,\ s_{l+1},\ \ldots,\ s_r)$, then the type of the k-th jade drawn by *Qingque* in the battle is $s_{l+((k-1) \bmod (r-l+1))}$. In addition, sometimes the server will update its jade sequence by changing the p-th jade to x.

Since there are many battles everyday, waiting for them all can be frustrating. Therefore, `AntiLeaf` requested your help. Given the initial sequence in the server and m events in order, each of which represents

either the server making an update or `AntiLeaf` encountering a battle, please help `AntiLeaf` to calculate the number of turns for each battle.

Input

The first line of input contains two positive integers n and m ($1 \le n, m \le 10^5$), representing the length of the initial sequence and the number of events, respectively.

The second line of input contains n integers separated by spaces, representing the initial sequence. Each of the n integers is either 1, 2, or 3, representing the three types of jades.

Then follow m lines. The i-th line of them represents an event, whose format is either of the following:

- 1 p_i x_i ($1 \le p_i \le n,\ x_i \in \{1, 2, 3\}$): the server makes an update, changing the p_i-th jade to x_i.
- 2 l_i r_i h_i a_i b_i ($1 \le l_i, r_i \le n.\ 1 \le h_i \le 10^{13},\ 0 \le a_i, b_i \le 10^9$): `AntiLeaf` encounters a battle, where the hidden draw pile corresponds with ($s_{l_i},\ s_{l_i+1},\ \ldots,\ s_{r_i}$), and the metadata of this battle is (h_i, a_i, b_i) – the HP of the enemy is h_i, and the damage when *Qingque* has four identical jades is a_i, or b_i otherwise.

Output

For each event of the second type, print an integer in a single line, representing the number of turns the battle takes. If the battle never ends, print -1 instead.

Example

standard input	standard output
5 5 1 2 3 1 2 2 1 1 10 1 0 1 2 3 2 2 4 10 2 1 1 3 2 2 1 5 100 5 2	30 8 44

Problem F. Collinear Exception

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

Do you know what is challenging for a geometry problem? Epsilons, NaNs, and even subnormal numbers that may affect your World Finals journey! (See "Subnormal behavior" section in ACM ICPC World Finals 2013 Solution sketches, Page 2)

So you dislike these floating-point issues in geometry problems. You decided to generate data containing only integral points, and moreover, ensured that no three points are collinear to avoid some divide-by-zero issues. How considerate of you!

However, everything comes with a price. It is frustrating to create a large dataset constrained within a small range — quite often, they end up with multiple collinear triples. Therefore, you give up and use a simple greedy strategy.

Consider an $n \times n$ grid where the coordinates along both the x-axis and the y-axis range from 1 to n. Within this grid, a permutation of all $n \times n$ integral points is given. Our objective is to greedily construct a maximal set S using the given permutation, so that no three points in S are collinear. We begin with an empty set and process each point in sequence. If the current point does not lie on any line formed by any two existing points in the set, it is added to the set; otherwise, it is skipped.

The approach is clear, but it still requires some coding work to get the job done. You need to write a program to report whether each insertion is successful.

Input

The first line contains a single integer n ($1 \le n \le 1000$).

For the next $n \times n$ lines, each contains two integers x_i, y_i ($1 \le x_i, y_i \le n$), indicating the i-th point.

It is guaranteed that the set of points includes each point from the set $\{1, \ldots, n\} \times \{1, \ldots, n\}$ exactly once.

Output

Output a `01` string of length $n \times n$, where the i-th character is `1` indicates whether the i-th point was successfully inserted into the set.

Example

standard input	standard output
3 1 1 1 2 1 3 2 1 2 2 2 3 3 1 3 2 3 3	110110000

Problem G. TNT Run

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Ah, the thrilling world of TNT Run! Let's dive into this explosive adventure, shall we?

Once upon a pixelated time, in the cubic realm of Minecraft, there existed a game called TNT Run. It was a game of strategy, speed, and survival, played on a two-dimensional plane. Our protagonist, Steve (because every Minecraft story needs a Steve), found himself at the starting point (x_s, y_s), ready to embark on a TNT-fueled escapade.

Initially, Steve is at a certain point (x_s, y_s), facing a direction(one of east, south, west, and north). In each second, Steve can either turn 90 degrees left or move 1 unit forward toward his current direction. The direction is similar to that in real-life. For example, let Steve be at $(2, 3)$, facing north. If he moves 1 unit forward, then he will be at $(2, 4)$. If he turns 90 degrees left, he will face west.

Also, there are additional rules about the game.

- Steve cannot turn left in his first or last operation.
- Steve cannot turn left twice consecutively.
- Steve cannot go to the same point twice.

Steve played this game thousands of times. Now, he is trying to remember some games. However, he only remembers the points he passed by and forgets the order. Finally, he finds it possible to seek help from your advanced programming skill.

The task is simple: given the points that Steve passed by, restore any possible moving sequence!

Input

The input consists of multiple test cases. The first line contains a single integer T ($1 \le T \le 2 \times 10^4$), representing the number of test cases. The description of the test cases follows.

The first line contains one integer n ($2 \le n \le 10^5$), the number of points Steve passed.

Each of the following n lines contains two integers x_i and y_i ($1 \le x_i, y_i \le 10^5$), representing the points Steve passed.

It's guaranteed that all n points are distinct.

It's guaranteed that the sum of n does not exceed $2 \cdot 10^5$.

Output

For each test case, output "-1"(without quote) if there is no possible solution, which means Steve remember the trial wrong.

Otherwise, output $|s|$ (the length of your string in the third line) in the first line.

Then, output three integers x_s, y_s and d in the second line. (x_s, y_s) is the initial point, and d indicates Steve's initial direction, where $d = 0, 1, 2, 3$ stands for east, north, west and south, respectively.

In the third line, output a 01 string s as Steve's moving sequence. If $s_i = 0$, Steve moves 1 unit forward in i-th operation. If $s_i = 1$, Steve turns left in i-th operation. If there are multiple solutions, you can output any.

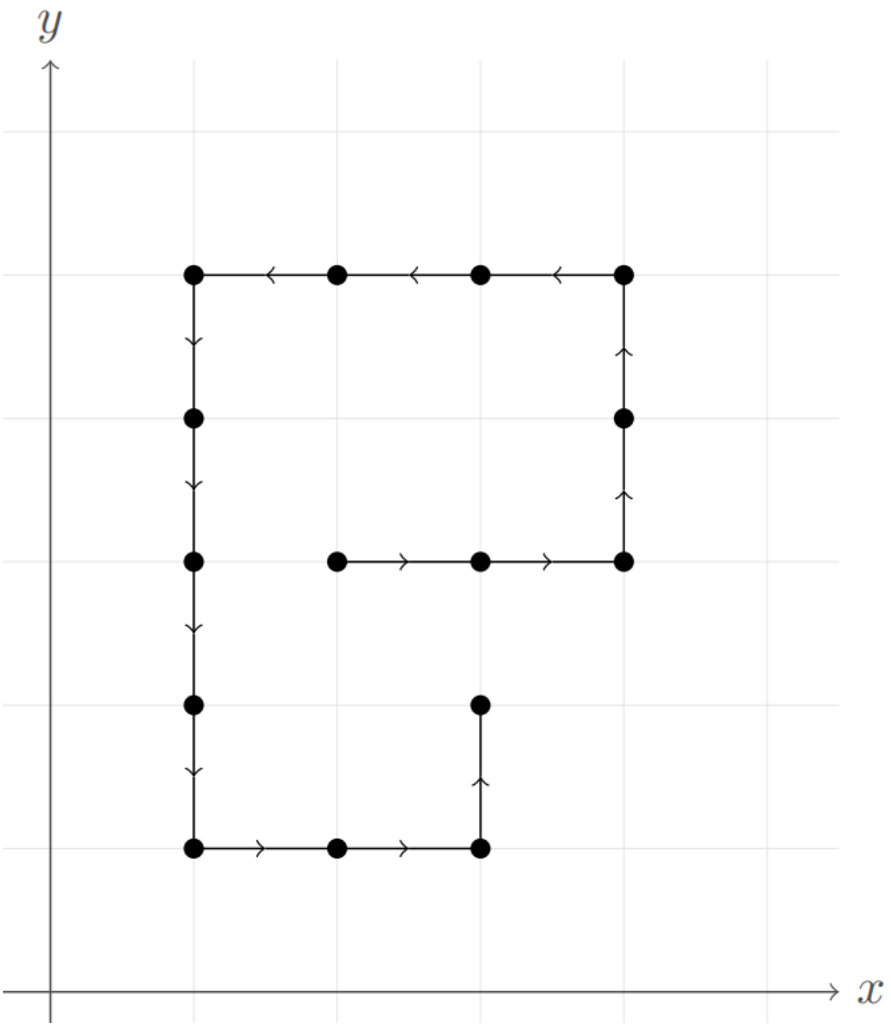
Note that Steve's moving sequence should satisfy all rules in the statement, that Steve must visit all points given in the input, and that Steve cannot visit those points not given in the input.

Example

standard input	standard output
4 9 1 1 1 2 1 3 2 1 2 2 2 3 3 1 3 2 3 3 6 1 1 1 2 1 3 2 2 2 3 2 4 15 1 1 2 1 3 1 1 2 3 2 1 3 2 3 3 3 4 3 1 4 4 4 1 5 2 5 3 5 4 5 8 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8	12 1 1 0 001001001010 -1 19 2 3 0 00100100010000010010 -1

Note

The following graph shows the third test case.


y
x

Problem H. All-in at the Pre-flop

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

In the backroom of the "All-in at the Pre-flop" casino, a crowd gathers nightly to witness a high-stakes duel that strips Texas Hold'em down to its rawest form of luck. Two players, known only by their monikers "Antileaf" and "Despradotwo", have become the main attraction. This evening, they bring their chips and a fierce desire to outlast the other, betting everything they have at the pre-flop stage in a daring display of brinkmanship. The crowd watches, mesmerized by the simplicity and intensity of the game where strategy is absent, and fortune rules. The entire pot goes to the winner, and the loser must leave with nothing.

In a simplified version of Texas Hold'em poker, two players face off in a duel where strategy is replaced by sheer luck. Each player starts with a fixed number of chips: the first player has a chips and the second player has b chips. In each game round, two players go all-in before even seeing the hole cards. After that, all community cards are dealt only once. Since nothing will change when the round is a tie, we assume that no tie happens, and the first player wins with probability 1/2 independently as it's a fair game. The round concludes as follows:

- If the winner in the round has at least the chips the loser has, the game ends and the winner takes all the chips.
- Otherwise, the loser pays the same amount of chips as the winner has, and another round starts afterwards.

You want to know the probabilities of each player winning the game (taking all the chips) under these conditions.

Input

The input consists of two integers, a, b ($1 \le a, b < 998\,244\,353, a+b < 998\,244\,353$), indicating the number of chips with which the first and second players start, respectively.

Output

Print two integers in a line separated by a space:

- The probability modulo 998 244 353, expressed as an integer, that the first player wins.
- The probability modulo 998 244 353, expressed as an integer, that the second player wins.

The expected value in question will be a rational number. If we express it as a fractional number $\frac{x}{y}$ where x and y are coprime positive integers, x will be coprime with $P = 998244353$. Therefore, print the only integer z between 0 and $P-1$ (inclusive) such that $xz \equiv y \pmod{P}$.

Example

standard input	standard output
1 3	748683265 249561089

Note

In this scenario, all possibilities conclude within two rounds. In the first round, if the first player loses, the game ends immediately. Otherwise, the next round starts with $a = 2, b = 2$, and the player who wins the second round takes all the chips. Therefore, it is only possible for the first player to win the entire game by winning both rounds, and the probabilities are 1/4 and 3/4.

Problem I. Riffle Shuffle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Riffle shuffling is a common method of shuffling playing cards, characterized by splitting the deck into two (in practice, roughly equal) parts and interleaving them together. Historically, riffle shuffling has been a focal point of interest because of its balance between being simple to perform and effective in randomizing a deck. This method has been meticulously analyzed by mathematicians and magicians alike to determine how many such shuffles are needed to achieve what is considered a "random" state of the deck. Persi Diaconis, a mathematician with a background as a professional magician, brought a significant insight into this study. His research highlighted that a deck of 52 cards, when given seven properly executed riffle shuffles, reaches a state close enough to random for practical purposes, such as in casino games. This finding bridges the gap between theoretical mathematics and practical application, providing a quantifiable goal for randomness through shuffling.

Today, your challenge is to present a side proof of the randomness provided by a riffle shuffle: if a shuffle is correct, i.e., close to a uniform random distribution, every permutation should be possible (and roughly equally likely) to reach. But how many steps are required to reach a specific permutation?

Given a deck of cards $1, \ldots, n$ ($2 \le n \le 52$) described by permutation p, devise a procedure to transform $1, \ldots, n$ into p using the minimum number of *riffle shuffles*.

To describe a *riffle shuffle*, a string t consisting of '`A`' and '`B`' of length n is used. For a deck described by permutation q,

- The deck is divided into two parts: the top s cards designated as a, and the remaining cards as b, where s represents the number of '`A`' characters in the string t.
- Repeat the following step n times: For the i-th operation, if the i-th character of t is '`A`', pop the front card from a; otherwise, pop it from b. Append this popped card to the end of the new deck q'.
- Update q to be q'.

Note that this definition is for the sake of clarity — in real life, the card is popped from the bottom of one deck, and prepended to the top of the new deck. However, two definitions are equivalent by reversing the operation string. Please, see the sample note section for an example.

Input

There are T ($1 \le T \le 10^5$) test cases in a single input file. For each test case, it consists of

- a line containing a single integer n ($2 \le n \le 52$), and
- a line containing a permutation of $1, \ldots, n$ separated by spaces, denoting the permutation p.

For each input file, it is guaranteed that the sum of n is not greater than 2×10^5.

Output

For each test case, print a single integer k ($0 \le k \le n$) in the first line, denoting the minimum number of operations, then for each of the k operations in order, print a string of length n consisting of only '`A`' and '`B`', which is able to uniquely determine the operation's steps.

Example

standard input	standard output
5 2 1 2 2 2 1 3 3 1 2 3 3 2 1 4 3 1 4 2	0 1 BA 1 BAA 2 BAA ABA 1 BABA

Note

For the last test case in the sample, $n = 4$ and the permutation $p = [3, 1, 4, 2]$.

To achieve this permutation with a single riffle shuffle, choose a split right before the third element (i.e., $i = 2$), the parts become:

- $a = [1, 2]$
- $b = [3, 4]$

Then, interleaving by choosing elements from b then a alternately (the 'BABA' pattern), results in:

- From b: 3
- From a: 1
- From b: 4
- From a: 2

Problem J. Doremy's Starch Trees

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

A rooted tree T_2 is called to be a unrooted tree T_1's starch tree if and only if these two conditions are satisfied at the same time (Denote that s_i is the set of all vertices in the i's subtree **on tree** T_2):

1. T_1 and T_2 have the same number of vertices.
2. $\forall$ vertices x that have child, $\forall$ x's child y, there is at least one vertex $z \in s_y$ connected to x **on tree** T_1.

Now Doremy has a unrooted tree X, and she remembers that a rooted tree Y is X's starch tree. However, Doremy forgets the root of Y. Can you help Doremy find the root or claim that there is no possible root?

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases. The description of the test cases follows.

The first line contains one integer n ($2 \le n \le 10^6$), the number of vertex in the tree.

The second line contains $n-1$ integers $a_2, \cdots, a_n$, indicating that there is an edge (i, a_i) on tree X. It is guaranteed that they form a tree.

The third line contains $n-1$ integers $b_2, \cdots, b_n$, indicating that there is an edge (i, b_i) on tree Y. It is guaranteed that they form a tree.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

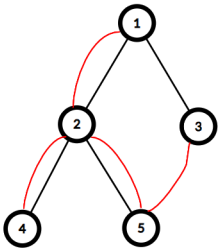
For each test case, output -1 if Y is not a starch tree of X. Otherwise, output a possible root r ($1 \le r \le n$). If there are multiple answers, output any.

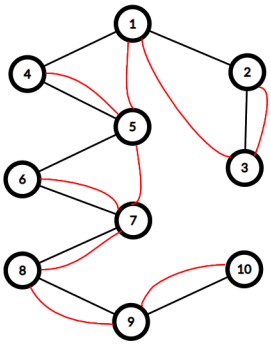
Example

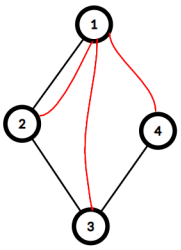
standard input	standard output
13 5 1 1 2 2 1 5 2 2 10 1 2 1 4 5 6 7 8 9 3 1 5 1 7 5 7 8 9 4 1 2 3 1 1 1 5 1 2 1 1 1 4 1 1 4 1 2 3 1 2 2 11 1 2 3 1 5 6 1 8 1 10 3 1 1 6 1 10 1 8 8 10 10 1 2 3 4 4 3 7 3 9 4 2 1 1 4 8 3 10 3 2 1 1 6 5 2 5 1 3 3 6 3 4 1 5 1 1 1 1 1 4 5 2 7 1 1 2 2 3 3 1 2 3 2 3 3 5 1 2 3 4 4 1 5 1 15 4 13 6 6 1 13 6 7 2 10 7 1 7 4 4 5 6 7 1 1 6 7 2 10 3 1 7 4	3 1 -1 3 4 -1 5 1 -1 3 4 -1 12

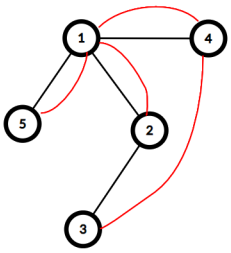
Note

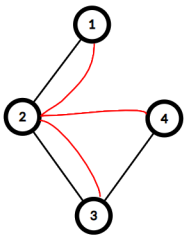
The following graph shows the first six test cases. Black edges belong to tree X, and red edges belong to tree Y.

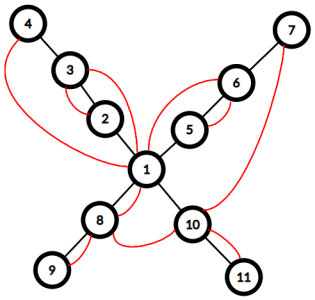

1
2
3
4
5


1
2
3
4
5
6
7
8
9
10


1
2
3
4


1
2
3
4
5


1
2
3
4


1
2
3
4
5
6
7
8
9
10
11

Problem K. Doremy's IQ 2

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	1024 megabytes

Doremy is asked to test n contests. The difficulty of contest i is a_i. In the following n days, Doremy has to test all n contests, one contest per day. Each contest should be tested only once, but can be tested in any order. Initially, Doremy's IQ is 0, and it may change after each test.

If Doremy chooses to test a contest with difficulty d and now Doremy's IQ is x, the following happens:

- if $d > x$, Doremy will feel inspired, so her IQ increases by 1;
- if $d < x$, Doremy will feel fooled, so her IQ decreases by 1;
- if $d = x$, Doremy will feel satisfied, and her IQ will not change.

Doremy wants to encounter as many $d = x$-type contest as possible. Please help Doremy find out the number.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains one integer n ($1 \le n \le 10^5$) — the number of contests.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-n \le a_1 \le a_2 \le \cdots \le a_n \le n$) — the difficulty of each contest.

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

Output

For each test case, output the maximum number of $d = x$-type contests that Doremy can encounter.

Example

standard input	standard output
5 5 1 2 3 4 5 5 -2 -1 0 1 2 8 -8 -3 -2 -2 -2 1 1 8 1 0 2 0 1	2 2 4 1 1

Problem L. Tada!

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

There is a combination lock with n ($1 \le n \le 5$) digits , each digit being between $0 \sim 9$. For each digit, turning it clockwise by one position will change the digit from x to $x+1$ mod 10. For example, 0 becomes 1, 1 becomes 2, and 9 turns into 0. Turning it counterclockwise does the opposite, changing the digit to $x+9$ mod 10.

Little Cyan Fish has locked the secrets of the China Clattering Processing Contest (CCPC) with this combination lock. Every day, to secure the secrets, Little Cyan Fish scrambles the combination by repeating the following operations for some number of times: turning an interval $[l, r]$ ($1 \le l \le r \le n$) of digits by one position, each turn accompanied by a distinct "tada" sound.

As Big Black Fish, who is secretly positioned next to Little Cyan Fish, you aim to steal the secrets for a reward. You eavesdrop on Little Cyan Fish's locking process daily, recording the number of "tada" sounds — which is the number of operations, and then inspect the digits on the lock in the dead of night.

You have collected m pieces of information, and you need to determine whether the information collected uniquely identifies the correct combination.

Input

The input consists of multiple test cases. The first line contains T ($1 \le T \le 1000$), the number of test cases.

For each test case, the first line contains two numbers, n, m ($1 \le n \le 5, 1 \le m \le 50$), separated by a space.

The next m lines contain a string of n digits s_i and an integer t_i ($0 \le t_i \le 50$), indicating the combination displayed is s_i, resulting from t_i operations on the correct combination. The digit string and the number are separated by a space.

It is guaranteed that $\sum_n 10^n \le 3 \times 10^5$ for all T cases.

Output

For each test case:

- If there is a unique solution, output a line with the n-digit combination.
- If there are multiple solutions, output a line with the string `MANY`.
- If there is no solution, indicating an error in the records, output a line with the string `IMPOSSIBLE`.

Example

standard input	standard output
3 3 3 003 1 003 3 025 1 3 2 000 1 999 1 1 2 0 0 1 0	014 MANY IMPOSSIBLE

Note

For the first sample, the following are possible operation sequences:

- $003 \xrightarrow{2\sim3,+} 014$;
- $003 \xrightarrow{2\sim3,+} 014 \xrightarrow{1\sim1,+} 114 \xrightarrow{1\sim1,-} 014$;
- $025 \xrightarrow{2\sim3,-} 014$.

For the second sample, $009, 099, 990, 900$ are all possible.

Problem M. Patchouli and Magic Formations

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	512 megabytes

Patchouli is an experienced magician who is good at manipulating the power of elements. Today, Patchouli is going to develop some new magic formations.

A magic formation consists of several element orbs, and the orbs are connected with some **directed** magic lines. The orb at the start of the magic line is called the predecessor, and the orb at the end of the magic line is called the successor.

There are m types of orbs, each with unlimited supply. Each type of orb has at most one predecessor, and the i-th type of orb has a_i **different** successor slots, each of which can be connected with **another** orb. In addition, the first successor slot of each type is privileged: it must not be empty, unless the orb has no successor.

Patchouli has not decided the number of orbs in the formation yet, so for each i between 1 and n (both inclusive), she wants to know how many different formations can be created with i orbs. The formations are not required to be connected.

Two formations are considered the same if and only if there is a bijection between the orbs of the two formations, such that the type of each orb and the existence of each magic line are preserved. Otherwise, they are considered different.

Input

Each input contains only one test case.

The first line of the input contains two integers n and m ($1 \leq n, m \leq 10^5$) separated by a space, representing the maximum number of orbs and the number of types of orbs, respectively.

The second line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \leq a_i \leq 10^5$), the i-th of which represents the number of **different** successor slots of the i-th type of orb.

Output

Output n space-separated integers in a single line, the i-th of which represents the number of different formations that can be created with i orbs, modulo 998244353.

Examples

standard input	standard output
3 2 1 2	2 10 42
3 3 1 2 3	3 21 174
10 4 2 3 5 7	4 36 836 20736 481300 10926348 247433796 628004857 283432588 490330663

Note

The output of the third sample is adjusted for the convenience of painting. In real test data it contains no extra line break.