Problem A. World Cup

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

The Chinese team is participating in the FIFA World Cup 4202, which includes 32 teams, and the Chinese team has number 1, each with a unique strength value a_i. Matches between any two teams will have a winner, which is the team with the higher strength value winning.

In the group stage, the 32 teams will be divided into 8 groups, each group consists of 4 teams and plays a round-robin tournament in which each team is scheduled for three matches against other teams in the same group. The number of winning matches is used to rank the teams in a group. The top two teams from each group will advance to the knockout stage.

The knockout stage is a single-elimination tournament in which teams play each other in one-off matches. It begins with the round of 16, in which the first place of each group plays against the second place of another group. This is followed by the quarter-finals, the semi-finals, and the final.

Specifically, denote A1 as the first place of group A, C2 as the second place of group C, and so on. The matches in the round of 16 are (1).A1 vs B2, (2).C1 vs D2, (3).E1 vs F2, (4).G1 vs H2, (5).B1 vs A2, (6).D1 vs C2, (7).F1 vs E2, (8).H1 vs G2.

Then, the matches of the quarter-finals are between (9).winners of (1) and (2), (10).winners of (3) and (4), (11).winners of (5) and (6), (12).winners of (7) and (8).

The semi-finals are between (13).winners of (9) and (10), (14).winners of (11) and (12).

And the final is between (15).winners of (13) and (14).

Given the strength of every team $a_1, \ldots, a_{32}$, suppose you can manipulate the grouping scheme, what is the best possible result for the Chinese team? Specifically, output

- `1` if the Chinese team wins the championship,
- `2` if the Chinese team loses in the final,
- `4` if the Chinese team loses in the semi-final,
- `8` if the Chinese team loses in the quarter-final,
- `16` if the Chinese team loses in the round of 16,
- `32` if the Chinese team does not advance to the knockout stage.

Input

Each test contains multiple test cases. The first line contains the number of test cases $t (1 \leq t \leq 10^3)$. The descriptions of the test cases follow.

The only line of a test case contains 32 different integers, indicating $a_1, \ldots, a_{32} (1 \leq a_i \leq 10^9)$.

Output

For each test case, print one line with one integer indicating the answer to the question.

Example

standard input	standard output
1 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1	1

Problem B. Graph

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

We call an undirected graph with n vertices "good" when for all $1 \le u, v \le n$, there exists at least one path between u, v, i.e., $u = x_1, x_2, \ldots, x_m = v$ such that $\gcd(u, v) = \gcd(x_1, x_2, \ldots, x_m)$.

We call an undirected graph with n vertices "perfect" when it is "good" and the number of edges in the graph is minimal. That is to say, any other "good" graph with n vertices has no less number of edges than this graph.

You need to count the number of "perfect" graphs with n vertices.

Since the answer could be very large, you only need to find it modulo $998\,244\,353$.

Input

The input contains only one single integer n $(2 \le n \le 10^{11})$ — the number of vertices in the "perfect" graph you need to count.

Output

The output contains only one single integer, representing the answer modulo $998\,244\,353$.

Example

standard input	standard output
4	8

Problem C. Permutation Counting 4

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

Given n pairs (l_i, r_i), you need to count how many permutations p of size n there are such that $l_i \le p_i \le r_i$. You only need to output the answer modulo 2.

Input

The input consists of multiple test cases. The first line contains a single integer t $(1 \le t \le 10^6)$ — the number of test cases. The description of the test cases follows.

The first line contains one single integer n $(1 \le n \le 10^6)$ — the size of the permutation p you need to count.

Then, the i-th line of the following n lines contains two integers l_i, r_i $(1 \le l_i \le r_i \le n)$.

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

Output

For each test case, output one single integer representing the answer modulo 2.

Example

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

Problem D. Protection War

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	1024 megabytes

The city of Country X is under attack by Country Y! To defend itself, Country X has set up n rows of fortifications, numbered consecutively from $1, 2, \ldots, n$. The military strength of the fortification numbered i is a_i. A fortification is considered **defeated** if its military strength is zero. Initially, no fortifications are **defeated**.

The battle lasts for a total of q days, and each morning, one of the following events occurs:

- Troop Movement: Given parameters x, y, the military strength of the fortification numbered x is set to y.
- Reinforcement: Given parameters l, r, v, the military strength of the fortifications numbered $l, l+1, \ldots, r$ is increased by v (including fortifications that are already **defeated**).
- Recruitment: The military strength of all fortifications that are not **defeated** is increased by 1.
- Training: Let $b_i = \max\{r - l + 1 \mid 1 \leq l \leq i \leq r \leq n, \text{s.t.} \forall l \leq j \leq r, a_j > 0\}$, which is the length of the longest continuous segment containing fortification i that has not been **defeated**. For all $1 \leq i \leq n$, set $a_i := a_i + b_i$.
- Parade: Given parameters l, r, inquire about the total military strength of the fortifications numbered $l, l+1, \ldots, r$.

Every evening, a rout event occurs: For the fortification numbered i, if $i < n$ and the fortification numbered $i+1$ has already been **defeated** by noon that day, then this fortification will also be **defeated** (i.e., its military strength becomes zero).

As the commander-in-chief of Country X, you need to provide the correct result for each "parade" operation.

Input

The first line contains two integers, n and q, representing the number of fortifications and the number of days, respectively.

The second line contains n integers, representing $a_1, \ldots, a_n$, the initial military strength of each fortification.

The next q lines describe the events that occur each morning for q days. In the i-th line, the first integer op indicates what event occurs on the i-th day:

- If $op = 1$, a troop movement occurs, and the parameters x, y are given next.
- If $op = 2$, a reinforcement occurs, and the parameters l, r, v are given next.
- If $op = 3$, a recruitment occurs.
- If $op = 4$, a training occurs.
- If $op = 5$, a parade occurs, and the parameters l, r are given next.

It is guaranteed that $1 \leq n, q \leq 3 \times 10^5$ and $1 \leq a_i \leq 10^5$. For the troop movement event, it is guaranteed that $1 \leq x \leq n$ and $0 \leq y \leq 10^5$. For the reinforcement event, it is guaranteed that $1 \leq l \leq r \leq n$ and $1 \leq v \leq 10^5$. For the parade event, it is guaranteed that $1 \leq l \leq r \leq n$.

Output

For each “parade” event, output one integer per line representing the answer.

Example

standard input	standard output
10 8 1 2 3 4 5 6 7 8 9 10 1 5 0 4 5 1 10 2 1 7 10 5 1 7 1 5 0 3 5 1 7	74 97 71

Problem E. Random Dungeon

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

Mike is playing a video game called Random Dungeon.

In this game, you will challenge a dungeon and get rewards based on your score every week.

Mike has discovered that the dungeon has N variations, numbered from 1 to N. Mike will challenge the dungeon N times, and for each challenge, the game chooses a variation that has not appeared in previous challenges with equal probability.

Mike will get a score of A_i challenging the dungeon variation i. He can choose to stop whenever he has completed a challenge, and his final score will be the score of the last challenge. At the end of the week, if his final score is x, he will be rewarded x coins. However, challenging the dungeon costs coins. Each challenge costs C coins.

If Mike acts to maximize the expected profit (coins earned at the end of the week minus coins spent on challenge), find the expected profit.

Input

The first line contains two integers N and C ($1 \le N \le 2 \cdot 10^5$, $1 \le C \le 10^9$).

The second line contains N integers $A_1, A_2, \dots, A_N$ ($1 \le A_i \le 10^9$) — the score of Mike challenging dungeon variation i.

Output

Output the expected profit. Your answer will be considered correct if the absolute error or the relative error does not exceed 10^{-9}. That is, if the correct answer is x, and your answer is y, your answer will be considered correct if $\frac{|x-y|}{\max\{1,|x|\}} \le 10^{-9}$.

Examples

standard input	standard output
3 1 1 2 3	1.1666666667
3 3 1 2 3	-1.0000000000
9 193138187 782710197 539624191 631858791 976609486 752268030 30225807 279200011 467188665 630132600	442999078.5373015873

Note

In the first example, the best strategy of Mike is: If the dungeon in the first challenge is variation 2 or 3, stop; otherwise, do a second challenge and stop. The expected profit is $\frac{1}{3} \cdot (2-1) + \frac{1}{3} \cdot (3-1) + \frac{1}{3} \cdot \left(\frac{2+3}{2} - 2\right) = \frac{7}{6}$.

In the second example, the best strategy of Mike is: do a single challenge and stop.

The third example contains extra line breaks to fit into the table.

Problem F. Make Max

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

You are given a sequence of n positive integers $[a_1, \ldots, a_n]$. You can apply the following operation to the sequence:

- Select a subarray $a[l \ldots r]$ $(1 \le l < r \le n)$ where not all elements are identical (i.e., there exist two integers $l \le i < j \le r$ such that $a_i \ne a_j$), and then change every element in this subarray to $\max_{l \le i \le r} a_i$.

Determine the maximum number of such operations that can be performed.

Input

Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 1000)$. Description of the test cases follows.

The first line of each test case contains an integer n $(1 \le n \le 2 \times 10^5)$.

The second line of each test case contains n integers $a_1, \ldots, a_n$ $(1 \le a_i \le 10^9)$.

The sum of n over all test cases does not exceed 4×10^5.

Output

For each test case, print the maximum number of operations that can be performed.

Example

standard input	standard output
4 2 1 2 2 2 2 7 1 1 1 2 2 2 2 3 1 2 3	1 0 3 3

Note

In the first test case, an optimal sequence of operations is:

1. Select $a[1 \ldots 2]$ and apply the operation, so that a becomes $[2, 2]$.

In the second test case, no operation can be performed.

In the fourth test case, an optimal sequence of operations is:

1. Select $a[1 \ldots 2]$ and apply the operation, so that a becomes $[2, 2, 3]$.
2. Select $a[2 \ldots 3]$ and apply the operation, so that a becomes $[2, 3, 3]$.
3. Select $a[1 \ldots 3]$ and apply the operation, so that a becomes $[3, 3, 3]$.

Problem G. the Median of the Median of the Median

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

Today is YQH's birthday, and she received a positive integer sequence of length n, denoted as $\{a_i\}_{i=1}^n$, as a gift.

YQH is very interested in medians, so she wants to find the median of the median of the median for this sequence. Specifically, let $b_{l,r}$ be the median of the multiset $\{a_i\}_{l\le i\le r}$, and let $c_{l,r}$ be the median of the multiset $\{b_{i,j}\}_{l\le i\le j\le r}$. Then, what YQH wants to find is the median of the multiset $\{c_{l,r}\}_{1\le l\le r\le n}$. However, she finds this task too difficult, so she asked you for help.

Note: If you are not familiar with the concept of a median, the median of a multiset of size m is the $\lceil m/2\rceil$-th smallest element in it.

Input

A single positive integer on the first line represents n.

The second line contains n positive integers representing $a_1, \ldots, a_n$.

It is guaranteed that $1 \le n \le 2000$ and $1 \le a_i \le 10^9$.

Output

Output a single line with one integer representing the answer.

Examples

standard input	standard output
4 1 3 1 7	1
8 3 3 8 4 5 3 8 5	4

Problem H. Rainbow Bracket Sequence

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

A bracket sequence is a string containing only characters `(` and `)`. A regular bracket sequence is a bracket sequence that can be transformed into a correct arithmetic expression by inserting characters `1` and `+` between the original characters of the sequence. For example, bracket sequences `()()` and `(())` are regular, and `)(`, `(`, and `)` are not.

There are m different colors, and you are given $\ell_1, \dots, \ell_m$. Consider bracket sequences of length $2n$, the colors of each position are $c_1, \dots, c_{2n}$. A regular bracket sequence is *colorful* if

- For all $i = 1, \dots, m$, denote t_i to be the number of positions of color i with left bracket `(` on it, then $t_i \geq \ell_i$ holds.

Given values of positions $v_1, \dots, v_{2n}$, the *value* of a regular colorful bracket sequence is the sum of values of the positions with left bracket.

You need to find the maximum value among all regular colorful bracket sequences. If there is no such sequence, output `-1` instead.

Input

Each test contains multiple test cases. The first line contains the number of test cases $t (1 \leq t \leq 100)$. The descriptions of the test cases follow.

Each test case contains four lines. The first line contains two integers $n, m (1 \leq n \leq 100, 1 \leq m \leq n)$, indicating the half of the length of the bracket sequence and the number of colors.

The second line contains m integers $\ell_1, \dots, \ell_m (0 \leq \ell_i \leq n)$, indicating the limit of each color. The third line contains $2n$ integers $c_1, \dots, c_{2n} (1 \leq c_i \leq m)$, indicating the color of each position. And the fourth line contains $2n$ integers $v_1, \dots, v_{2n} (1 \leq v_i \leq 10^9)$, indicating the value of each position.

It is guaranteed that the sum of n over all test cases does not exceed 500.

Output

For each test case, print one line with one integer indicating the answer to the question or `-1` if there is no possible sequence.

Example

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

Note

In the first example, sequence `()(())` gets the maximum 9. And in the second example, there is no possible sequence since there are only 3 left brackets.

Problem I. Boxes

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

Given n points in three-dimensional space, the task is to partition them into some mutually disjoint subsets $S_1, S_2, \ldots, S_k$. For any $i \neq j$, the partition must satisfy at least one of the following three conditions:

1. $\text{volume}(\text{conv}(S_i) \cap \text{conv}(S_j)) = 0$,
2. $\text{conv}(S_i) \subseteq \text{conv}(S_j)$,
3. $\text{conv}(S_j) \subseteq \text{conv}(S_i)$.

Here, the convex hull $\text{conv}(S_i)$ of a subset S_i is defined as:

$$\text{conv}(S_i) = \left\{ \sum_{p \in S_i} \lambda_p p \mid \lambda_p \geq 0, \sum_{p \in S_i} \lambda_p = 1 \right\}.$$

The goal is to maximize

$$6 \sum_{i=1}^{k} \text{volume}(\text{conv}(S_i)).$$

The challenge is to find the optimal partition that achieves the maximum total volume of the convex hulls while ensuring these constraints are met.

Input

There are multiple test cases in a single test file. The first line of the input contains a single integer T ($1 \leq T \leq 3000$), indicating the number of test cases.

For each test case, the first line of the input contains one integer n ($4 \leq n \leq 3000$) — the number of points. The following n lines each contain three integers x_i, y_i, and z_i ($0 \leq x_i, y_i, z_i \leq 10^6$), representing the coordinates of point p_i in three-dimensional space.

It's guaranteed that no four points are coplanar, and the sum of n over all test cases does not exceed 3000.

Output

For each test case, output a single integer — the maximum sum of volumes of the convex hulls under the given conditions, multiplied by 6. It can be proven that this value is always an integer.

Example

standard input	standard output
2 4 0 0 1 0 0 2 0 1 1 1 1 1 10 2 6 3 2 9 0 2 1 0 3 7 3 0 5 6 10 9 2 4 4 2 8 5 2 4 6 9 6 7 5	1 943

Problem J. Rivals

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

You are playing a game. There are n enemies in this game. The health point of the i-th enemy is a_i, and the first c enemies are "key" enemies.

There are k rounds in the game. In each round, let S be the enemies with health point a_i greater than 0 (i.e. $S = \{i | 1 \le i \le n \land a_i > 0\}$), and you will choose one enemy from the set S with the same probability (i.e. for each i in S, you will choose it with probability $\frac{1}{|S|}$) and decrease its health point by 1 (i.e. let a_i be $a_i - 1$).

If after k rounds, the health point of every "key" enemy has been decreased to 0, you win the game; otherwise, you lose the game.

Now for each k such that $1 \le k \le \sum_{i=1}^{n} a_i$, you need to count the probability of winning the game after k rounds. You only need to output the answer modulo $998\,244\,353$.

Input

The first line contains two integers n, c ($1 \le c \le n \le 30$) — the number of enemies and "key" enemies.

The second line contains n integers, the i-th integer of which is a_i ($1 \le a_i \le 10$) — the health point of the i-th enemy in the beginning.

Output

The output contains $\sum_{i=1}^{n} a_i$ integers. The k-th integer represents the probability of winning the game after k rounds modulo $998\,244\,353$.

Examples

standard input	standard output
`5 3` `1 1 1 1 1`	`0 0 299473306 199648871 1`
`8 5` `3 5 3 2 2 5 4 4`	`0 0 0 0 0 0 0 0 0 0 0 0 0 0 851829480 293319617 60309444`

Problem K. AC Automation Chicken

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

Braised Chicken loves Aho-Corasick automatons. Therefore, he came up with the following problem related to Aho-Corasick automatons. Before we dive into the problem, he kindly reminds you of the following definitions:

- A **Trie** T is a rooted tree, on each edge of which a character is written. A vertex x on the Trie should not have two children y and z such that the characters written on the edges (x, y) and (x, z) are the same.
- Suppose there is a given Trie T rooted at r. For a node x, **the string represented by** x is the resultant string when you concatenate the characters on the edges that are on the path in T from r to x, in the order they appear on the path. Particularly, the string represented by r is the empty string. It can be proved that no two different vertices represent equal strings.
- We say a string S **exists** in a Trie T if and only if there exists a vertex x in T such that the string represented by x is S.
- A **fail tree** F of a given Trie T is a rooted tree whose root is the root of T, r. Define S_x as the string represented by node x. For a non-root node x, suppose U is the longest proper suffix of S_x (a proper suffix of a string S is a suffix of S that is not equal to S) that exists in T. Then, $fail_x$ is defined as the vertex of T such that $S_{fail_x} = U$. Note that the empty suffix of S_x always exists in T, so $fail_x$ always exists. The edge set of F is $\{(x, fail_x) \mid x \in [1, n], x \neq r\}$. It can be proven that these edges form a tree.

Braised Chicken has a Trie T of n vertices, numbered 1 through n. You do not know its root. The character set is all the integers in $[1, n]$. Then, he built the corresponding fail tree F of the Trie. After that, he combined the edges of T (directed away from the root) and the edges of F (directed towards the root) to get a **unweighted directed graph** G with n nodes and $2n - 2$ directed edges. To be more specific, G is constructed as follows:

- For every non-root vertex u, G contains an edge $fa_u \to u$ where fa_u is u's father on T.
- For every non-root vertex u, G contains an edge $u \to fail_u$.

And finally comes your task: given the unweighted directed graph G of n vertices (the edges are given in an arbitrary order), you need to

- find out the root of T, vertex r;
- identify which edges of G are on T and which edges are on F;
- find a valid way of writing a character (an integer in $[1, n]$) on each edge of T so that T and its corresponding fail tree indeed accord with G.

If there is no valid way of assigning the root and information of the edges, you should also report this fact.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5 \times 10^4$). Description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 10^5$).

Then $2n-2$ lines follow. Each of these $2n-2$ lines contains two integers x, y ($1 \le x, y \le n$), denoting an edge in G that goes from x to y.

The sum of n over all test cases does not exceed 5×10^5.

Output

For each test case, if there is no valid way of assigning the root and information of the edges, print `No`.

Otherwise, print `Yes` in the first line. Then print $n-1$ lines. Each of these $n-1$ lines should contain three integers x, y, z ($1 \le x, y, z \le n$), denoting that T has an edge from x to y (x should be the father of y) with the character z written on it. The vertex without any incoming edges is the root of T. It should hold that the edges you output form a rooted Trie corresponding with G.

You can print the edges in any order. If there are multiple ways of assigning T, print any.

Example

standard input	standard output
4 4 1 4 4 1 1 2 2 1 2 3 3 4 2 1 2 1 2 1 3 1 2 2 1 2 3 3 2	Yes 1 2 1 2 3 2 4 1 1 No Yes Yes 1 2 1 2 3 1

Note

In the first test case, the root of T is 4. It can be verified that $fail_1 = 4, fail_2 = 1, fail_3 = 4$.

Note that there are also other valid outputs for this input: for example, letting 1 be the root and T have edges $(1 \to 2, 1), (2 \to 3, 2), (1 \to 4, 2)$ (here, $(x \to y, z)$ means that the father of y is x, and character z is written on the edge from x to y) is also a valid answer.

Problem L. Bull Farm

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

Arthur owns a bull farm. His business is doing so well that he has recently started to struggle to provide an adequate number of cattle stalls. There are n stalls on his farm. He also has $(n-1)$ bulls. It is important that no two bulls share a stall (otherwise they could hurt each other). Arthur has a remote control which allows him to move the bulls between stalls. Pressing a button results in moving all the bulls simultaneously. In particular, if a bull was in the j-th stall before pressing the i-th button, it moves to the $t_{i,j}$-th stall.

Arthur has to make repairs in some of the boxes, so he prepared a list of q queries. Each query asks if it is possible to manoeuvre the bulls in such a way that at the end there is no bull in the b-th stall, if at the beginning only the a-th stall was empty and one can use only the first c buttons. Remember that no two bulls can share a stall at any moment!

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 2000$) denoting the number of test cases. The description of test cases follows.

Each test case begins with a line containing three integers n, ℓ, q ($1 \le n, \ell \le 2000$, $1 \le q \le 10^6$), the number of stalls, the number of buttons of the remote control and the number of queries.

In the i-th of the following ℓ lines there are n numbers $t_{i_1}, t_{i_2}, \ldots, t_{i_n}$, where t_{i_j} is the number of the stall to which the j-th bull will move after pressing the i-th button.

The i-th of the following q lines contains three numbers a_i, b_i, c_i, the parameters of the i-th query.

The numbers t_{i_j} and a_i, b_i, c_i are encoded to decrease the input size. The code of a number x is a two letter word consisting of ASCII characters $48 + \lfloor x/50 \rfloor$ and $48 + (x \bmod 50)$. The code of the three numbers is the concatenation of their codes. E.g. the word ?;;=EL encodes three numbers 761 563 1078.

It is guaranteed that the sum of n over all test cases does not exceed 2000, the sum of ℓ over all test cases does not exceed 2000 and the sum of q over all test cases does not exceed 10^6.

It is guaranteed that $1 \le t_{i_j}, a, b \le n, 0 \le c \le \ell$.

Output

For each test case, print a sequence of q characters in a new line. The i-th character should be equal to 1 if it is possible to safely move the bulls in the i-th query. It should be equal to 0 otherwise.

Examples

standard input	standard output
2 5 2 4 0305040201 0404040404 030300 020500 050102 020501 6 2 4 030603010601 010203060504 030202 060402 050602 060401	1011 0100
1 3 3 6 020202 030301 030201 020102 030203 010201 010303 020303 010202	010101

Problem M. Find the Easiest Problem

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

You are given all the submissions from an ICPC-style competition. For each submission, you know the team name, the problem ID, and whether the submission was accepted or rejected. Your task is to determine which problem was the easiest.

You may not be familiar with the ICPC rules, so here's a brief explanation:

- Each team can make multiple submissions for a problem. A team is considered to have solved a problem if at least one of their submissions for that problem is accepted.
- The number of teams that solved a problem is the number of distinct teams that have at least one accepted submission for that problem.

We define the easiest problem in the competition as the one that has been solved by the most teams. If there are multiple problems with the same number of teams solving them, the problem with the lexicographically smallest ID is considered the easiest.

Two teams are considered different if and only if their team names are different. Similarly, two problems are considered different if and only if their problem IDs are different.

Input

The first line contains an integer t ($1 \le t \le 1000$), the number of test cases.

For each test case:

The first line contains an integer n ($1 \le n \le 10^5$), the number of submissions.

Each of the next n lines contains a team name, a problem ID, and the result of the submission.

- The team name is a non-empty string consisting of at most 10 characters, which can include both uppercase and lowercase English letters.
- The problem ID is a single uppercase English letter ('A' to 'Z').
- The result of the submission is either `accepted` or `rejected`.

It is guaranteed that there is at least one `accepted` submission in every test case.

It is also guaranteed that the total number of submissions across all test cases does not exceed 10^5.

Output

For each test case, output a single line with the problem ID of the easiest problem.

Example

standard input	standard output
2 5 teamA A accepted teamB B rejected teamC A accepted teamB B accepted teamD C accepted 4 teamA A rejected teamB A accepted teamC B accepted teamC B accepted	A A