Problem A. City

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Hi ICPCer, welcome to Xi'an.

Being a beautiful ancient city, Xi'an is the capital city of Zhou, Qin, Han, and Tang Dynasties. With a long history, the streets in Xi'an have a grid pattern.

Attracted by the streets' structure, Coach *Pang* would like to conduct his research on them. He draws an $n \times m$ grid on the board. The grid consists $n+1$ vertical line segments and $m+1$ horizontal line segments. The vertical and horizontal line segments intersect at exactly $(n+1) \times (m+1)$ points, forming $n \times m$ unit squares. We call the $(n+1) \times (m+1)$ intersections *grid point*s. Output the number of line segments(not only vertical or horizontal) l satisfying the following three conditions:

1. The length is not zero.
2. Both endpoints of l are grid points.
3. The midpoint of l is a grid point.

Input

The only line contains two integers $n, m (1 \le n, m \le 1000)$.

Output

Print the answer in a single line.

Examples

standard input	standard output
1 1	0
2 3	14

Problem C. Dirichlet k-th root

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Mathematician Pang learned Dirichlet convolution during the previous camp. However, compared with deep reinforcement learning, it's too easy for him. Therefore, he did something special.

If $f, g : \{1, 2, \ldots, n\} \to \mathbb{Z}$ are two functions from the positive integers to the integers, the Dirichlet convolution $f * g$ is a new function defined by:

$$(f * g)(n) = \sum_{d|n} f(d)g(\frac{n}{d}).$$

We define the k-th power of an function $g = f^k$ by

$$f^k = \underbrace{f * \cdots * f}_{k \text{ times}}.$$

In this problem, we want to solve the inverse problem: Given g and k, you need to find a function f such that $g = f^k$.

Moreover, there is an additional constraint that $f(1)$ and $g(1)$ must equal to 1. And all the arithmetic operations are done on $\mathbb{F}_p$ where $p = 998244353$, which means that in the Dirichlet convolution, $(f * g)(n) = \left(\sum_{d|n} f(d)g(\frac{n}{d})\right) \bmod p$.

Input

The first line contains two integers n and k $(2 \le n \le 10^5, 1 \le k < 998244353)$.

The second line contains n integers $g(1), g(2), ..., g(n)$ $(0 \le g(i) < 998244353, g(1) = 1)$.

Output

If there is no solution, output -1.

Otherwise, output $f(1), f(2), ..., f(n)$ $(0 \le f(i) < 998244353, f(1) = 1)$. If there are multiple solutions, print anyone.

Example

standard input	standard output
5 2 1 8 4 26 6	1 4 2 5 3

Problem E. Flow

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

One of *Pang*'s research interests is the maximum flow problem.

A directed graph G with n vertices is *universe* if the following condition is satisfied:

- G is the union of k vertex-independent simple paths from vertex 1 to vertex n of the same length.

A set of paths is vertex-independent if they do not have any internal vertex in common.

A vertex in a path is called internal if it is not an endpoint of that path.

A path is simple if its vertices are distinct.

Let G be a *universe* graph with n vertices and m edges. Each edge has a non-negative integral capacity. You are allowed to perform the following operation any (including 0) times to make the maximum flow from vertex 1 to vertex n as large as possible:

Let e be an edge with positive capacity. Reduce the capacity of e by 1 and increase the capacity of another edge by 1.

Pang wants to know what is the minimum number of operations to achieve it?

Input

The first line contains two integers n and m ($2 \le n \le 100000, 1 \le m \le 200000$).

Each of the next m lines contains three integers x, y and z, denoting an edge from x to y with capacity z ($1 \le x, y \le n$, $0 \le z \le 1000000000$).

It's guaranteed that the input is a *universe* graph without multiple edges and self-loops.

Output

Output a single integer — the minimum number of operations.

Examples

standard input	standard output
4 3 1 2 1 2 3 2 3 4 3	1
4 4 1 2 1 1 3 1 2 4 2 3 4 2	1

Problem G. Happiness

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

Pang has graduated from college 3 years and he really misses the time he spent with ICPC (Interspecies Collegiate Pokemon Camp).

There are 10 problems in one contest in ICPC. n participating teams have 300 minutes to solve them. After the contest, teams are ranked according to the most problems solved. Teams who solve the same number of problems are ranked by least total time. The total time is the sum of the time consumed for each problem solved. The time consumed for a solved problem is the time elapsed from the beginning of the contest to the submittal of the first accepted run plus 20 penalty minutes for every previously rejected run for that problem. There is no time consumed for a problem that is not solved. If two teams tie, their *solution time lists* are calculated. A team's solution time list is a list consisting of solution times of all problems solved by that team, sorted in descending order. The solution time of one problem is the time elapsed from the beginning of the contest to the submittal of the first accepted run of that problem. (We do not add a penalty for the solution time.) The team with a lexicographically smaller solution time list has a better rank. A list $(a_1, \ldots, a_k)$ is lexicographically smaller than $(b_1, \ldots, b_k)$ if there exists an integer $i \in [1, k]$ such that $a_i < b_i$ and $a_j = b_j$ for all integers $j \in [1, i)$. If teams still tie, *Pang*'s team is assumed to have a better rank.

After determining the rank, prizes will be awarded. Initially, a team with rank r will get $\lfloor 5000/r \rfloor$ happiness. Then medals are awarded: Teams with rank 1 to $\lfloor n/10 \rfloor$ are awarded gold medal. The *happiness* of receiving a gold medal is 1200. Teams with rank $\lfloor n/10 \rfloor + 1$ to $3\lfloor n/10 \rfloor$ are awarded silver medal. The *happiness* of receiving a silver medal is 800. Teams with rank $3\lfloor n/10 \rfloor + 1$ to $6\lfloor n/10 \rfloor$ are awarded bronze medal. The *happiness* of receiving a bronze medal is 400. In addition to medals, for each problem, the team solved it first gets 800 happiness. The team with at least one solution and the smallest solution time overall teams and all problems gets an extra 700 happiness. The team with at least one solution and the largest solution time overall teams and all problems gets an extra 500 happiness. In the case of a tie, *Pang*'s team can always get happiness.

There were n teams in a contest *Pang* participated. He remembers all the submissions (time and verdict) of all other teams. For each problem, he also remembers if he knew the solution to that problem and the number of rejected runs and times he needed to solve it.

If *Pang* solved problems in the wisest order, what is the maximum happiness he could get? Note that *Pang* cannot solve any problem after 300 minutes from the beginning of the contest (He can solve problems at exactly 300 minutes). Once *Pang* solves a problem, he needs to submit it immediately and solve another one. He can't postpone his submission to get the last submission happiness.

Input

The first line contains an integer n denoting the number of teams ($10 \le n \le 300$, n is a multiple of 10).

Each of the next $n - 1$ lines describes one team and contains the statuses of the 10 problems. For each problem, if it is not solved by the team, the status contains a single character "-". Otherwise, the status contains two integers t and w separated by a single space denoting the solution time and the number of rejected runs before the solution time ($1 \le t \le 300, 0 \le w \le 10$). Statuses of different problems are separated by ",".

The last line describes *Pang*'s team. For each problem, if *Pang* did not know how to solve it, the status contains a single character "-". Otherwise, the status contains two integers x and y separated by a single space denoting the required time and the number of rejected runs before *Pang* could solve it ($1 \le x \le 300, 0 \le y \le 10$). Statuses of different problems are separated by ",".

There are no extra spaces and other characters in the statuses of *Pang* and other teams.

Output

Output one integer — the maximum happiness.

Example

standard input	standard output
10 233 1,-,-,7 7,257 4,173 5,117 1,-,-, 85 3 -,231 0,167 0,257 7,-,-,122 4,283 0, 215 4,- 41 1,-,290 8,-,-,-,-,246 7,120 3,184 9 142 8,243 7,69 0,-,41 9,-,279 1,264 4,-,74 9 53 8,-,187 9,60 1,48 8,99 10,-,-,55 7,259 5 250 0,-,-,-,166 0,16 3,-,82 4,73 0, 184 3 -,-,-,-,105 3,-,-,-,152 4,- -,84 5,98 8,-,120 8,241 3,94 1,-,28 7,109 8 280 6,246 5,58 9,-,-,-,-,-,-,- 38 10,-,227 10,187 9,182 1,-,203 9 ,254 7,-,-	1800

Note

Note that the sample input and sample output contain wrapped lines to fit in the width of page.

Problem H. King

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

As we all know, the number of *Pang*'s papers follows exponential growth. Therefore, we are curious about *King* sequence.

You are given a prime p. A sequence $(a_1, a_2, \ldots, a_n)$ is a *King* sequence if and only if there is an integer $1 \le q < p$ such that for all integers $i \in [2, n]$, $qa_{i-1} \equiv a_i \pmod{p}$.

Given a sequence $B = (b_1, \ldots, b_m)$, what is the length of the longest *King* subsequence of B?

A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

Pang is super busy recently, so the only thing he wants to know is whether the answer is greater than or equal to $\frac{n}{2}$.

If the length of the longest *King* sequence is less than $\frac{n}{2}$, output -1. Otherwise, output the length of the longest *King* subsequence.

Input

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

The first line in a test case contains two integers n and p ($2 \le n \le 200000$, $2 \le p \le 1000000007$, p is a prime). The sum of n over all test cases does not exceed 200000.

The second line in a test case contains a sequence $b_1, \ldots, b_n$ ($1 \le b_i < p$).

Output

For each test case, output one line containing the answer which is -1 or the length of the longest *King* subsequence.

Example

standard input
4 6 1000000007 1 1 2 4 8 16 6 1000000007 597337906 816043578 617563954 668607211 89163513 464203601 5 1000000007 2 4 5 6 8 5 1000000007 2 4 5 6 7

standard output
5 -1 3 -1

Problem J. Permutation

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You are given a permutation $p_1, p_2, \ldots, p_n$. You can do the following operations repeatedly:

- Choose an interval $p_l, p_{l+1}, \ldots, p_{l+c} (l \geq 1, l+c \leq n)$ where p_l is the smallest element in this interval, you can permutate $p_{l+1}, \ldots, p_{l+c}$ in arbitrary way.
- Choose an interval $p_l, p_{l+1}, \ldots, p_{l+c} (l \geq 1, l+c \leq n)$ where p_{l+c} is the smallest element in this interval, you can permutate $p_l, \ldots, p_{l+c-1}$ in arbitrary way.

You want to know how many distinct permutations you can get using operations. The answer can be large, output the answer modulo 998244353.

Input

The first line contains an integer T denoting the number of test cases ($1 \leq T \leq 100000$).

The first line in a test case contains two integers n and c ($2 \leq c \leq 500000$, $2 \leq n \leq 500000$). The sum of n over all test cases does not exceed 500000.

The second line in a test case contains a permutation $p_1, \ldots, p_n$ ($1 \leq p_i \leq n$).

Output

For each test case, output one line containing the answer modulo 998244353.

Example

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

Problem M. Value

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Pang believes that one cannot make an omelet without breaking eggs.

For a subset A of $\{1, 2, \ldots, n\}$, we calculate the score of A as follows:

1. Initialize the score as 0.
2. For any $i \in A$, add a_i to the score.
3. For any pair of integers (i, j) satisfying $i \geq 2$, $j \geq 2$, $i \in A$ and $j \in A$, if there exists positive integer $k > 1$ such that $i^k = j$, subtract b_j from the score.

Find the maximum possible score over the choice of A.

Input

The first line contains a single integer n ($1 \leq n \leq 100000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 1000000000$).

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq 1000000000$).

Output

Print a single integer x — the maximum possible score.

Examples

standard input	standard output
4 1 1 1 2 1 1 1 1	4
4 1 1 1 1 1 1 1 2	3

Problem N. Natural Splice Point

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Define the balance of an array of integers to be the difference between its max and min. Given an array of integers, you are to splice it into two nonempty arrays and compute the balance of each. You have found the *natural splice point* when the largest balance of the two subarrays is minimized.

Put another way, given an array $A[1 \ldots N]$, find the natural splice point x, $1 \leq x < N$, such that $\max(\max(A[1 \ldots x]) - \min(A[1 \ldots x]), \max(A[x+1 \ldots N]) - \min(A[x+1 \ldots N]))$ is minimized.

Input

Each line of input presents a single test case: a positive integer $N \leq 400\,000$, denoting the size of the array, followed by N positive integers, each less than $2 \cdot 10^6$. If $N = 1$, then this is a signal that the input has ended; this is not a test case and should not be processed.

Output

For each test case, determine the value of x, the natural splice point. In the case where several natural splice points exist, report the smallest index.

Example

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

Problem O. Orientation

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You have a list of points in the coordinate plane which represent the planned bike race. The riders have to start and end at the same point.

Check if the points in the given order form left- or right- handed orientation.

Input

The first line of the input is the number K of input data sets ($1 \le K \le 22$), followed by the K data sets, each of the following form:

The first line of each data set contains the number of points N, $3 \le N \le 100$. The next N lines each contain two integers x and y, a coordinate of next point, where x is plotted on the x-axis (from left to right), y on the y-axis (downside up), and $100 \le x, y \le 100$. Taken in order, the N points will always form a closed, non-overlapping polygon, where no three consecutive points are collinear.

Output

Print "`RIGHT`", when looking down on the plane, rider would have his right hand touching the interior of the polygon as he traverses its points in the input order. Otherwise, output "`LEFT`.

Examples

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

Problem P. Pocket Calculator

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Peter so bored that he sums up integers from 1 to N on a somewhat ancient pocket calculator, one number at a time. That is, he typed '1", then pressed the '+' button, then typed '2", then pressed the '+' button, then typed '3', and so on. But he is so bored so he can have skipped one number less than N.

Given the final sum S, can you determine what was the missing number?

Input

You will be presented with several thousand test cases composed only of nonnegative integers, one per line. Each integer is the final sum S, where $S < 2^{56}$. The input ends when S = 0. This is not a test case and should not be processed.

Output

For each test case, output the missing number on a line by itself. If no number was missing, output -1 instead.

Example

standard input	standard output
44 45 46 0	1 -1 9

Problem Q. Quest for Food

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Traveling sometimes is tiring and can really work up an appetite. Luckily, there is a huge variety of exotic foods to be found in the cities on the way

Unfortunately, there is not enough space in the your car to store a refrigerator, so if you want to eat an exotic food, you must eat it right away. But, If you eat, you will become very full and will not be able to eat again while visiting the immediately following city.

Each food has a perceived value, and your goal is to maximize the total sum value for all the foods he actually eats. The order in which you visit the cities has already been set and cannot be changed. By selectively choosing which foods to eat, what is the maximum total sum value you can achieve?

Input

The first line is the number K of input data sets ($1 \le K \le 25$), followed by the K data sets, each of the following form: The first line of each data set contains an integer n indicating the number cities on your way, where $1 \le n \le 5 \cdot 10^4$. Next follows one line containing n integers v_i, $1 \le v_i \le 1000$, indicating the values of the foods in the cities. The order of v_i in the list is also the order in which you will encounter them.

Output

For each test case, output the maximum sum of values you can achieve.

Example

standard input	standard output
2 3 3 8 4 4 12 8 9 10	8 22

Problem R. Robot and Blocks

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You have a robot designed to collect blocks as quickly as possible.

Robot works in $M \times N$ room. Each cell in the room is one of:

- '.' — Empty space
- 'B — Block
- 'S — Starting location

There will be exactly one starting location. Robot can move North, East, South or West by one cell in one second. Robot can carry at most one block at any time. If robot is not carrying a block and enters a cell with a block, it can choose to pick it up instantly.

If robot is carrying a block, it cannot enter a cell with a block. When robot is holding a block and enters the cell containing the starting location, the block is instantly transported to the storage, and robot will be free to collect another block.

Your task is to find minimum time for robot to collest all the blocks.

Input

The first line of input is an integer T ($1 \le T \le 100$). T test cases follow. Each test case begins with a line containing two integers, M and N, the dimensions of the room ($1 \le M, N \le 500$). Next come M lines of N characters representing the room layout, as described above.

Output

For each case, print one integer — the minimum number of seconds it will take robot to collect all the blocks.

Example

standard input	standard output
1 3 3 BBB .S. B.B	18

Problem S. Secret Message System

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Two of your friends Bill and Jeff have developed a secret system for sending messages to each other.

The encrypted message consists of one or more words, only consisting of lowercase English characters '`a`'-'`z`', seperated by spaces.

Each word corresponds to one character, '`a`'-'`z`' or space, and is found by adding the value of each character in the word and taking the remainder when dividing by 27. '`a`' is assigned the value 0, '`b`' is assigned the value 1 and so on up to '`z`' which is assigned the value 25. Based on the calculated remainder the values 025 corresponds to '`a`'-'`z`' as before, and the value 26 corresponds to a space.

Help your friends and write a program which takes a coded message and outputs the decoded message.

Input

The first line of the input consists of a single integer T ($1 \le T \le 100$), the number of test cases. Each test case consists of a single line containing a secret message. The secret message contains only lower case characters and spaces, and there are only spaces between words and never two or more consecutive spaces. Total length of the message does not exceed 1000 characters.

Output

For each test case, output the corresponding decoded message.

Example

standard input	standard output
2 a b c d e f xx yy zz	a b c d e f tvx