The 3rd Universal Cup

Stage 15: Chengdu

November 2-3, 2024

This problem set should contain 13 problems on 17 numbered pages.

Based on

International Collegiate Programming Contest (ICPC)

Prepared & Hosted by

Problem A. Arrow a Row

Time limit: 1 second
Memory limit: 1024 megabytes

Define an "arrow string" as a string that meets the following conditions:

- The length of the string is at least 5.
- The string starts with `>` and ends with `>>>`.
- The rest of the string consists only of `-`.

For example, `>-->>>` and `>--->>>` are valid arrow strings, while `>->>` and `>->->>>` are not.

Sauden gives you a string s of length n, consisting of `>` and `-`. You need to create s by performing a series of painting operations on a string of the same length n that consists entirely of `*`. In one painting operation, you can choose a substring of length at least 5 and transform it into an arrow string. The total number of operations you perform cannot exceed n.

If it is impossible to obtain the string s using no more than n painting operations, output `No`. Otherwise, output `Yes` and provide the details of the painting operations. If there are multiple solutions, output any.

Input

The first line contains a single integer T ($1 \le T \le 10^4$), indicating the number of test cases.

Each test case contains a string s of length n ($5 \le n \le 10^5$) in a single line, consisting only of `>` and `-`.

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

Output

For each test case, if the given string cannot be obtained by performing no more than n painting operations, output `No` in a single line. Otherwise, output `Yes` and a positive integer m ($1 \le m \le n$), which denotes the number of painting operations to perform. Then output m lines, each contains two integers p ($1 \le p \le n-4$) and l ($5 \le l \le n+1-p$), indicating the starting position of the selected substring and its length.

Example

standard input	standard output
4 >>->>> >>>-> >>>>> >->>>>>>	Yes 2 1 5 2 5 No No Yes 2 2 7 1 5

Note

For the fourth test case in the example, the painting process is shown below::

******** → *>--->>> → >->>>>>>

Problem B. Athlete Welcome Ceremony

Time limit: 2.5 seconds
Memory limit: 1024 megabytes

Chengdu is about to host the 2025 World Games. During the athlete welcome ceremony at the opening event, there will be n volunteers dressed in one of three different types of traditional folk costumes, lined up to welcome the athletes. These costumes are denoted by type `a`, `b`, and `c`. The positions of the volunteers have been determined, and now we need to assign costumes to the volunteers. To achieve a specific visual effect, adjacent volunteers must not wear the same type of costume.

Among the n volunteers, some already have one of the three types of folk costumes, while others do not have any and require custom-made costumes provided by the organizers. There are Q custom-making plans, each specifying: making x sets of costumes `a`, y sets of costumes `b`, and z sets of costumes `c`.

For each custom-making plan, determine how many different valid costume arrangements can be made after distributing the custom-made costumes to the volunteers who do not have any costumes. Specifically, determine the number of ways for assigning costumes `a`, `b` and `c` under the condition of not exceeding the limits of the given plan. If two arrangements differ in the type of costume assigned to the same volunteer, they are considered different. Note that the same type of costumes are not distinguished from each other. As the number may be very large, please output the answer modulo $10^9 + 7$.

Input

The first line contains two integers n ($1 \le n \le 300$) and Q ($1 \le Q \le 10^5$), representing the number of volunteers and the number of custom-making plans, respectively.

The second line is a string s of length n. It is guaranteed that the string s contains only the characters `a`, `b`, `c`, and `?`. If the i-th character is one of `a`, `b`, and `c`, it indicates that the i-th volunteer already has the corresponding costume; otherwise, if it is `?`, it indicates that the i-th volunteer does not have any costume.

Each of the next Q lines contains three integers x, y, z ($0 \le x, y, z \le 300$), representing a custom-making plan. It is guaranteed that the sum $x + y + z$ is no less than the number of volunteers without costumes.

Output

Output Q lines, with the i-th line containing an integer that represents the number of valid costume arrangements that satisfy the requirements for the i-th custom-making plan. Please output the answer modulo $10^9 + 7$.

Examples

standard input	standard output
6 3 a?b??c 2 2 2 1 1 1 1 0 2	3 1 1
6 3 ?????? 2 2 2 2 3 3 3 3 3	30 72 96

Note

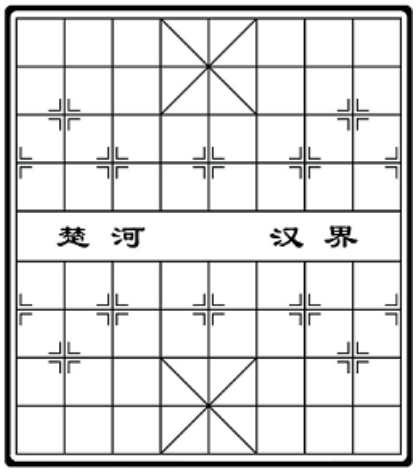
In the first sample, the valid costume arrangements for the first custom-making plan are `acbabc`, `acbcac`, and `acbcbc`.

Problem C. Chinese Chess

Time limit: 1 second
Memory limit: 1024 megabytes

This is an interactive problem.

Chinese chess is a two-player strategy board game that is very popular in China. There are 10 rows and 9 columns on a Chinese chess board. We denote the point on the r-th row from bottom to top and on the c-th column from left to right by (r, c) ($0 \le r \le 9$, $0 \le c \le 8$).



There are six types of pieces in this problem: kings, mandarins, rooks, knights, bishops, and pawns. Note that cannons are not considered. Additionally, all pieces can be moved anywhere on the board, which differs slightly from the original rules. The movement rules for each piece are as follows (assuming the current position is (r, c)):

Chess	Symbol	Possible positions in one move
King	J	$(r+1, c)$ $(r-1, c)$ $(r, c+1)$ $(r, c-1)$
Mandarin	S	$(r+1, c+1)$ $(r+1, c-1)$ $(r-1, c+1)$ $(r-1, c-1)$
Rook	C	all (r', c') such that $r' = r$ or $c' = c$ except (r, c) itself
Knight	M	$(r+2, c+1)$ $(r+2, c-1)$ $(r-2, c+1)$ $(r-2, c-1)$ $(r+1, c+2)$ $(r+1, c-2)$ $(r-1, c+2)$ $(r-1, c-2)$
Bishop	X	$(r+2, c+2)$ $(r+2, c-2)$ $(r-2, c+2)$ $(r-2, c-2)$
Pawn	B	$(r+1, c)$ if $r \le 4$, otherwise $(r+1, c)$ $(r, c+1)$ $(r, c-1)$

Of course, pieces cannot move outside the board's boundaries. Now that you understand the rules, let's play a game.

I have placed a piece on the board, and you cannot see its position or type. However, I will provide a set A of n possible positions where the piece might be located. Your goal is to determine the type of the piece using the minimum number of queries. For each query, you can select a position on the board, and I will tell you the minimum number of moves required for the piece to move to that position, or I will tell you if it is impossible for the piece to reach that position.

Interesting, isn't it? What is even more interesting is that I am actually gaming against you in secret. Truth be told, although set A is fixed, I have not fixed the type and position of this piece from the start. My strategy is to adjust the answer to each of your queries in a way that maximizes the number of queries you make, without contradicting the information already given. In the case where both of our strategies are optimal, your number of queries m is actually determinable. I am sure you are smart enough to figure it out. Let the game begin!

Interaction Protocol

Firstly, you should read an integer n ($1 \le n \le 90$) from standard input, indicating the size of the set A.

Then, read n lines. Each line contains two integers r and c, indicating a position in the set A. All positions are guaranteed to be distinct.

After reading these $n+1$ lines, you should output an integer m to standard output, indicating the number of queries. If your output m is incorrect, you will get a `Wrong answer` verdict. You must then make exactly m queries.

To make a query, output `? r c` ($0 \le r \le 9, 0 \le c \le 8$) in a single line. Then you should read an integer from standard input, indicating the minimum number of moves required for the piece to reach the selected position, or -1 if the piece cannot move to that position. If your program makes an invalid query or exceeds the number of queries, the interactor will terminate immediately, and you will receive a `Wrong answer` verdict.

To output your guess about the type of the piece, you need to output `! ch`, where `ch` means the symbol representing the piece that is shown in the statement.

After you make a query or output your answer, including the m and the type of the piece, do not forget to output the End of Line `\n` and flush the output. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;

Examples

standard input	standard output
1 9 0 6	 1 ? 3 6 ! S
4 2 1 2 3 2 5 2 7 3 1	 2 ? 0 0 ? 2 0 ! J

Note

For the first example, the number of moves required for each of the six types of pieces to move from position $(9, 0)$ to $(3, 6)$ is as follows:

King (J)	Mandarins (S)	Rooks (C)	Knights (M)	Bishops (X)	Pawns (B)
12	6	2	4	3	-1

Thus, you only need to make one query to determine the answer.

Problem D. Closest Derangement

Time limit: 3 seconds
Memory limit: 1024 megabytes

Blackbird has a permutation p of length n. He wants to find a derangement q of p, which means q is another permutation of length n satisfying $q_i \neq p_i$ for each $i = 1, 2, \ldots, n$. At the same time, $\sum_{i=1}^{n} |p_i - q_i|$ should be minimized. A permutation q that satisfies the above conditions is called the closest derangement of p.

There may be multiple closest derangements of p, and your task is to output the k-th smallest closest derangement in lexicographical order. If there are fewer than k closest derangements of p, output -1.

A permutation of length n refers to a sequence of length n where all elements are distinct and are positive integers from 1 to n. Permutations can be sorted in lexicographical order. Let a and b be two distinct permutations of length n. Then, $a < b$ if and only if at the smallest index i where $a_i \neq b_i$, it holds that $a_i < b_i$.

Input

The first line contains an integer T ($1 \leq T \leq 10^4$), representing the number of test cases.

For each test case, the first line contains two positive integers n ($2 \leq n \leq 2 \cdot 10^5$) and k ($1 \leq k \leq 10^9$). The second line contains n positive integers $p_1, p_2, \ldots, p_n$, representing the permutation p.

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

Output

For each test case, if there are at least k closest derangements, output n positive integers $q_1, q_2, \ldots, q_n$ in a single line separated by spaces, representing the k-th smallest closest derangement of p in lexicographical order. Otherwise, output -1.

Example

standard input	standard output
2 2 2 2 1 3 2 1 2 3	-1 3 1 2

Note

For the first test case, $[1, 2]$ is the only closest derangement, so output -1.

For the second test case, $[2, 3, 1]$ and $[3, 1, 2]$ are closest derangements of p, and $[3, 1, 2]$ is larger than $[2, 3, 1]$ in lexicographical order.

Problem E. Disrupting Communications

Time limit: 2 seconds
Memory limit: 1024 megabytes

The enemy has established communication outposts across several locations, which can be represented as nodes and edges in a network. This network forms a tree — a type of graph that is connected and has no cycles. As a specialist in communications engineering, your task is to disrupt their communications.

Each communication occurs along a simple path between two nodes in the tree. You have the ability to select a subgraph of this tree and disrupt every node in that subgraph. If the communication path includes a disrupted node, the communication is successfully disrupted. The subgraph you select must consist of a subset of nodes and edges from the original tree, and it must be connected, meaning it is also a tree.

The communication network consists of n nodes, labeled from 1 to n. Your mission involves answering q separate queries. For each query, you are given two nodes, u_i and v_i, and you must determine how many different subgraphs you can select to disrupt the communication between the two nodes. Since the number may be very large, you should provide the answer modulo 998244353. It is possible that $u_i = v_i$, which indicates an internal communication within a node, and you are also able to disrupt it by selecting a subgraph that contains the node.

Input

The first line contains a single integer T ($1 \le T \le 10^4$), indicating the number of test cases.

The first line of each test case contains two integers n ($2 \le n \le 10^5$) and q ($1 \le q \le 10^5$), denoting the number of nodes and the number of queries.

The second line contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), indicating that nodes i and p_i are connected by an edge.

Each of the next q lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n$), representing the i-th query.

It is guaranteed that the sum of n and the sum of q over all test cases do not exceed $3 \cdot 10^5$, respectively.

Output

For each test case, output q lines, each containing the result modulo 998244353 for one of the q queries.

Example

standard input	standard output
2 3 2 1 1 2 3 1 2 5 3 1 1 2 2 4 5 2 4 2 3	6 5 14 13 15

Note

In the first case of the example, 6 connected subgraphs can be selected in total. For the first query, all of them can disrupt the communication. For the second query, 5 of them can disrupt the communication; the exception is the subgraph consisting only of node 3.

Problem F. Double 11

Time limit: 6 seconds
Memory limit: 1024 megabytes

With the Double 11 Shopping Festival approaching, a store is managing its best-selling products for the event.

There are n types of best-selling products, and the daily sales volume for each type is s_i. Products need to be replenished multiple times to meet sales demand, and due to limited warehouse space, the total inventory must not exceed the storage capacity.

If only one type of product is replenished at a time, the workload becomes too high. To optimize replenishment, the store decides to divide the product types into m groups, assigning a replenishment parameter to each group. For group j, the replenishment parameter k_j is a positive real number, which means for each product type i in this group, a volume of $k_j \cdot s_i$ will be prepared for each replenishment, and it will be replenished $\frac{1}{k_j}$ times per day on average. Note that k_j can be either greater than, less than or equal to 1.

Let c_i be the group index of product type i. The maximum inventory in the warehouse can be represented by $\sum_{i=1}^{n} k_{c_i} \cdot s_i$. Due to the warehouse capacity limitation, this value cannot exceed 1.

Your task is to find a scheme that divides the n product types into m groups and assigns a replenishment parameter to each group such that the total number of replenishments per day is minimized while satisfying the warehouse capacity limitation, i.e., minimize $\sum_{i=1}^{n} \frac{1}{k_{c_i}}$, subject to $\sum_{i=1}^{n} k_{c_i} \cdot s_i \le 1$. For convenience, you should output the square root of the minimal answer, i.e., $\sqrt{\sum_{i=1}^{n} \frac{1}{k_{c_i}}}$. Note that you can assign the same replenishment parameters to different groups.

Input

The first line contains two integers n and m ($1 \le m \le n \le 2 \cdot 10^5$), indicating the number of product types and the number of groups.

The second line contains n integers s_i ($1 \le s_i \le 10^5$), indicating the sales volume of product type i.

Output

Output one real number, indicating the answer. Your output will be considered correct if the relative or absolute error does not exceed 10^{-9}.

Examples

standard input	standard output
4 2 1 2 3 4	6.1911471295571
10 3 1 2 3 4 5 6 7 8 9 10	22.5916253665141

Note

For the first example, let $k_1 = \frac{1}{3+\sqrt{21}}, k_2 = \frac{1}{7+\sqrt{21}}$, and $c_1 = c_2 = 1, c_3 = c_4 = 2$. We will get the maximum inventory as $(1+2)k_1 + (3+4)k_2 = 1$, and the total number of replenishments is $\frac{2}{k_1} + \frac{2}{k_2} = 20 + 4\sqrt{21}$. Therefore, the answer is $\sqrt{20+4\sqrt{21}} \approx 6.1911471295571$, which is the minimum.

Problem G. Expanding Array

Time limit: 2 seconds
Memory limit: 1024 megabytes

Given an integer array $a_1, a_2, \ldots, a_n$ of length n, you can perform any number of operations on this array. In each operation, you can choose two adjacent elements a_i and a_{i+1} ($1 \le i < n$), and insert one of the following three values between them: a_i `and` a_{i+1}, a_i `or` a_{i+1}, or $a_i \oplus a_{i+1}$. Your task is to determine the maximum number of distinct values that can exist in the array after performing any number of operations.

Note: x `and` y represents the bitwise AND of x and y. x `or` y represents the bitwise OR of x and y. $x \oplus y$ represents the bitwise XOR (exclusive OR) of x and y.

Input

The first line contains a single integer n ($2 \le n \le 10^5$), representing the length of the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$), representing the elements of the array.

Output

Output a single integer, representing the maximum number of distinct values that can be obtained in the array after performing any number of operations.

Examples

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

Problem H. Friendship is Magic

Time limit: 1 second
Memory limit: 1024 megabytes

Rockdu is a pony living in Ponyville. His best friend, Macaronlin, also lives there. Rockdu values friendship so much that he shares everything with Macaronlin, even integers.

Here comes the question: how to share an integer with someone else? For an integer x, Rockdu splits it into two parts. Specifically, he treats x in decimal form as a string without leading zeros, and splits it into two non-empty substrings at a certain position. He then considers these two substrings as two separate decimal integers, denoted as x_1 (the former) and x_2 (the latter).

Rockdu wants x_1 and x_2 to be as close in value as possible. Therefore, among all possible splits, he chooses the one that minimizes the absolute difference between x_1 and x_2. For example, if $x = 1003$, there are three possible ways to split it: 1|003, 10|03, and 100|3. Rockdu chooses the first way because it yields the smallest absolute difference: $|1 - 3| = 2$, whereas the other ways give $|10 - 3| = 7$ and $|100 - 3| = 97$.

Let $f(x)$ be defined as the smallest absolute difference between the two integers resulting from splitting x. For example, $f(1003) = 2$. Given two integers l and r, Rockdu wants to calculate the sum of $f(i)$ for all i in the range $[l, r]$. Since the answer may be very large, please output it modulo $10^9 + 7$.

Input

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

Each test case contains two integers l, r ($10 \le l \le r \le 10^{18}$) in a single line.

Output

For each test case, output the answer modulo $10^9 + 7$ in a single line.

Example

standard input	standard output
2 108 112 114514 1919810	31 86328270

Note

For the first test case in the sample:

- $f(108) = \min(|1 - 8|, |10 - 8|) = \min(7, 2) = 2$
- $f(109) = \min(|1 - 9|, |10 - 9|) = \min(8, 1) = 1$
- $f(110) = \min(|1 - 10|, |11 - 0|) = \min(9, 11) = 9$
- $f(111) = \min(|1 - 11|, |11 - 1|) = \min(10, 10) = 10$
- $f(112) = \min(|1 - 12|, |11 - 2|) = \min(11, 9) = 9$

Therefore, $\sum_{i=108}^{112} f(i) = 2 + 1 + 9 + 10 + 9 = 31$.

Problem I. Good Partitions

Time limit: 1 second
Memory limit: 1024 megabytes

Lawliet has a sequence of numbers of length n, denoted as $a_1, a_2, \ldots, a_n$, and he wants to determine how many good partitions exist.

A partition size k is considered a good partition size if it satisfies $1 \leq k \leq n$ and, after dividing the sequence a into parts by partition size, each resulting sub-sequence is non-decreasing. The partitioning method is as follows:

- The sequence a is divided into $\lceil \frac{n}{k} \rceil$ parts.
- For the i-th part ($1 \leq i \leq \lceil \frac{n}{k} \rceil - 1$), the elements are $a_{(i-1)\times k+1}, a_{(i-1)\times k+2}, \ldots, a_{i\times k}$.
- For the $\lceil \frac{n}{k} \rceil$-th part, the elements are $a_{(\lceil \frac{n}{k} \rceil - 1)\times k+1}, \ldots, a_n$. Note that the length of the last part may be less than k.

Lawliet finds this problem too simple, so he will make q modifications. Each modification provides two positive integers p and v, indicating that the value of a_p will be changed to v.

Your task is to help Lawliet calculate the number of good partition sizes before any modifications and after each modification.

Input

The first line contains an integer T ($1 \leq T \leq 10$), representing the number of test cases.

For each test case, the first line contains two integers n ($1 \leq n \leq 2 \cdot 10^5$) and q ($1 \leq q \leq 2 \cdot 10^5$), representing the length of the sequence and the number of modifications.

The second line contains n integers, representing the sequence $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 2 \cdot 10^9$).

The following q lines each contain two integers p ($1 \leq p \leq n$) and v ($1 \leq v \leq 2 \cdot 10^9$), indicating that the element at the p-th position in the sequence will be modified to v.

It is guaranteed that the sum of n and the sum of q over all test cases do not exceed $2 \cdot 10^5$, respectively.

Output

For each test case, output $q + 1$ lines, representing the number of good partition sizes before any modifications and after each modification.

Example

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

Note

Initially, the only good partition size is $k = 1$.

After the first modification, the sequence becomes $[4, 5, 2, 6, 1]$. Both $k = 1$ and $k = 2$ are good partition sizes.

After the second modification, the sequence becomes $[4, 5, 5, 6, 1]$. The good partition sizes are $k = 1$, $k = 2$, and $k = 4$.

Problem J. Grand Prix of Ballance

Time limit: 2 seconds
Memory limit: 1024 megabytes

Ballance is a classic game where players use the keyboard to control a ball through complex structures high above the ground, avoiding falls while solving puzzles to reach the end of each level. Recently, the player community has developed online mods and hosts regular online competitions, such as the Grand Prix of Ballance.

The Grand Prix consists of n levels, numbered from 1 to n, with m participants, numbered from 1 to m. The competition uses a point system: players earn points based on their ranking in each level, and the sum of their points across all levels determines the final standings. Each level has a designated start time, and participants must complete the level as quickly as possible. As a staff member, you receive a server log during the match containing three types of messages (it is guaranteed that $1 \le x \le n$ and $1 \le id \le m$):

- `1 x` — Type 1: the match on Level x has started.
- `2 id x` — Type 2: participant indexed id has completed Level x.
- `3 id x` — Type 3: participant indexed id voluntarily gives up completing Level x.

A Type 1 message indicates the start of Level x until a new Type 1 message appears for a different level. Only messages that correspond to the currently active level are considered valid; all messages for other levels should be ignored. Messages before the first Type 1 message are also ignored. Each level appears at most once in a Type 1 message.

Each player may yield multiple Type 2 and Type 3 messages per level, but only the first valid message counts. Specifically:

- Messages are ignored if they do not match the active level indicated by the Type 1 message.
- If a player's first valid message for a level is Type 2, they are considered to have completed the level successfully at that moment, and the player's any subsequent messages for that level are ignored.
- If a player's first valid message for a level is Type 3, they are considered to have given up completing the level at that moment, and the player's any subsequent messages for that level are ignored.
- If a player yields no messages for a level, they are considered not to have completed the level.

Points are awarded to participants who complete the level as follows: the first player to complete the level receives m points, the second receives $m - 1$ points, and so on. Participants who do not complete the level, including those who give up, receive no points.

Your task is to output the current total points of each participant in descending order based on the log. If multiple participants have the same points, they should be listed in ascending order by their index.

Input

The first line contains an integer T ($1 \le T \le 10^4$), indicating the number of test cases.

For each test case, the first line contains three integers n, m, and q ($1 \le n \le 10^5$, $2 \le m \le 10^5$, $1 \le q \le 2 \cdot 10^5$), indicating the number of levels, participants, and log messages, respectively.

The following q lines contain the log messages as specified above. Of course, the messages are presented in chronological order. The log may not contain all levels, as you may receive it midway through the competition. You only need to process the current results.

It is guaranteed that the sum of n, the sum of m, and the sum of q over all test cases do not exceed $5 \cdot 10^5$, respectively.

Output

For each test case, output m lines, each containing two integers: id and x, where id is the participant's index and x is their total points, sorted in descending order of points. If multiple participants have the same points, list them in ascending order by their index.

Example

standard input	standard output
3 3 4 6 1 2 2 1 1 2 2 2 3 3 2 2 3 2 2 1 2 3 4 8 1 2 2 1 1 2 2 2 3 3 2 2 3 2 2 1 2 1 1 2 1 1 3 4 7 1 2 2 1 1 2 2 2 3 3 2 2 3 2 2 1 2 1 1	2 4 1 3 3 0 4 0 1 7 2 4 3 0 4 0 2 4 1 3 3 0 4 0

Problem K. Magical Set

Time limit: 1 second
Memory limit: 1024 megabytes

You have a magical set that initially contains n distinct integers. You discover that these numbers can generate energy by dividing into their factors. In each step, you can select any number greater than 1 from the set, remove it, and insert one of its factors. The factor you insert must not be equal to the original number. Additionally, due to the instability of the magical set, your operation must ensure that the numbers in the set remain distinct.

Each operation generates one unit of energy, and your goal is to maximize the total energy generated by performing as many operations as possible. Given the initial numbers in the set, determine the maximum amount of energy that can be generated, i.e., the maximum number of operations that can be performed.

Input

The first line contains an integer n ($1 \le n \le 300$), indicating the number of integers in the initial set.

The second line contains n distinct integers a_i ($1 \le a_i \le 10^9$), representing the numbers in the initial set.

Output

Output a single integer indicating the maximum amount of energy that can be generated, i.e., the maximum number of operations that can be performed.

Examples

standard input	standard output
3 2 4 6	3
6 2 3 5 6 10 12	3

Problem L. Recover Statistics

Time limit: 1 second
Memory limit: 1024 megabytes

You recently conducted a survey on how much time university students spend commuting from their dorms to school buildings. You believe that this survey could significantly improve campus planning, making commuting easier for both students and faculty. As part of your analysis, you calculated the P50, P95, and P99 commute times to support your conclusions. Here, Px commute time being y means that **exactly** $x\%$ of the commute times in the entire dataset are less than or equal to y. For example, the P50 of the set $\{1, 1, 4, 5, 1, 4\}$ can be 1, 2 or 3, since there are **exactly** $6 \times 50\% = 3$ values less than or equal to 1, 2 or 3. However, there are no valid P95 or P99 for these values because $6 \times 95\%$ and $6 \times 99\%$ are not integers.

Unfortunately, something went wrong — you accidentally deleted the entire dataset. The only values you have left are the P50, P95, and P99 of the commute times. Since you do not have time to redo the survey, you need to reconstruct a set of data that matches all of the P50, P95, and P99 values.

Input

The input consists of three lines. The first line contains a single integer a, representing the P50 value. The second line contains a single integer b, representing the P95 value. The third line contains a single integer c, representing the P99 value. ($1 \le a < b < c < 10^9$)

Output

Output two lines. The first line should contain a single integer n ($100 \le n \le 10^5$), representing the length of the data set.

The second line should contain n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), representing the reconstructed data set. The integers can be in any order. Any output that meets the requirements will be considered correct.

Example

standard input	standard output
50 95 99	100 1 2 ... 100

Note

The example output omits the values between 3 and 99 (inclusive) for display purposes. In your output, you must include all of these values.

Please note that the definition of Px values in this problem is different from the x-th percentile.

Problem M. Two Convex Holes

Time limit: 5 seconds
Memory limit: 1024 megabytes

Consider two opaque planes $z = z_1$ and $z = z_2$ ($z_1 < z_2$) in a three-dimensional Cartesian coordinate system. Each plane has a convex polygonal hole, denoted as P_1 and P_2 respectively, which allows light to pass through.

There is a point light source L moving in the plane $z = z_0$ ($z_2 < z_0$), in a direction parallel to the xOy plane. The light source starts at the initial position (x_0, y_0, z_0) at time $t = 0$ and moves with a constant velocity $\boldsymbol{v} = (v_x, v_y, 0)$. Therefore, at any time t, the position of the light source is given by $(x_0 + v_x t, y_0 + v_y t, z_0)$.

For a point A in the plane $z = 0$, define it as **illuminated** at time t if and only if the segment LA intersects the interiors (including boundaries) of both polygons P_1 and P_2. The **illuminated area** at time t, denoted as $f(t)$, is the area formed by all illuminated points in the plane $z = 0$.

Define the **average illuminated area** over the time period $[t_1, t_2]$, denoted as $\mathbb{E}[f(t)|t \sim U(t_1, t_2)]$, as the expected value of $f(t)$ over the interval $[t_1, t_2]$, assuming t is a uniformly distributed random variable over $[t_1, t_2]$.

Given multiple time periods $[t_1, t_2]$, your task is to find the average illuminated area for each of these periods.

Input

The first line contains a single integer T ($1 \le T \le 10^4$) indicating the number of test cases.

For each test case, the first line contains five integers x_0, y_0, z_0, v_x, v_y ($-10^5 \le x_0, y_0 \le 10^5$, $1 \le z_0 \le 10^5$, $-10^3 \le v_x, v_y \le 10^3$), representing the initial position of the light source (x_0, y_0, z_0) and its velocity vector $\boldsymbol{v} = (v_x, v_y, 0)$. It is guaranteed that $\boldsymbol{v} \neq (0, 0, 0)$.

The second line contains two integers n_1 and z_1 ($3 \le n_1 \le 10^5$, $1 \le z_1 \le 10^5$), indicating the number of vertices of polygon P_1 and the value of z_1. Each of the following n_1 lines contains two integers x_i and y_i ($-10^5 \le x_i, y_i \le 10^5$), describing the vertices of P_1 in counterclockwise order.

The following line contains two integers n_2 and z_2 ($3 \le n_2 \le 10^5$, $1 \le z_2 \le 10^5$), indicating the number of vertices of polygon P_2 and the value of z_2. Each of the following n_2 lines contains two integers x_j and y_j ($-10^5 \le x_j, y_j \le 10^5$), describing the vertices of polygon P_2 in counterclockwise order.

It is guaranteed that no three or more vertices are collinear for P_1 and P_2.

The following line contains an integer q ($1 \le q \le 10^5$), indicating the number of queries. Each of the following q lines contains two integers t_1 and t_2 ($0 \le t_1 \le t_2 \le 10^3$), representing a time period.

It is guaranteed that the sum of $n_1 + n_2$ and the sum of q over all test cases do not exceed 10^5, respectively, and $z_1 < z_2 < z_0$.

Output

For each query, output a real number representing the average illuminated area. Your answer will be considered correct only if the relative or absolute error between your answer and the correct answer does not exceed 10^{-4}.

Example

standard input	standard output
1 0 0 3 0 -1 4 1 1 0 3 0 3 2 1 2 4 2 0 0 1 0 1 1 0 1 3 0 10 1 2 1 1	0.450000000 1.125000000 2.250000000

Note

For the example, the projections of convex polygons P_1 and P_2 onto the xOy plane at $t = 0$, and the movement of these projections, are illustrated below. Polygon $A_1B_1C_1D_1$ is the projection of polygon P_1, and polygon $A_2B_2C_2D_2$ is the projection of polygon P_2.