Problem A. Water

Time limit	1000 ms
Mem limit	262144 kB
OS	Windows

Now there are n people lining up to get some water, the i^{th} person wants to get c_i liters of water into his or her bottle.

When the bucket of water on the water dispenser runs out (Each bucket of water provides C liters of water) , the one who is using the water dispenser (includes the last one) will replace the bucket on the dispenser with a new bucket or water, even if he or she have got enough water. Then he or she leaves right away, so one may leaves with less water than one wants.

Now you want to know how many buckets of water needed (including the first bucket).

Input

The first line contains a single integer T $(1 \leq T \leq 100)$ - the number of testcases.

Each testcase contains two lines.

The first line contains two integers n, C $(1 \leq \sum n \leq 10^6, 1 \leq C \leq 10^3)$.

The second line contains n integers c_i $(1 \leq c_i \leq 10^3)$.

Output

For each testcase, print a positive integer representing the number of buckets needed.

Examples

Input	Output
4 3 1 6 4 5 4 1 1 6 4 5 2 10 5 2 5 10 8 6 7 10 2	4 5 1 3

Problem B. Avoiding the Abyss

Time limit	1000 ms
Mem limit	1048576 kB
OS	Windows

You are standing on a point with integer coordinates (x_s, y_s). You want to walk to the point with integer coordinates (x_t, y_t). To do this, you can walk along a sequence of line segments. But there is a swimming pool in your way. The swimming pool is an axis aligned rectangle whose lower left corner is on the point (x_l, y_l) and the upper right corner is on the point (x_r, y_r). You cannot ever cross the swimming pool, not even on the border. However, it is dark and you do not know the coordinates (x_l, y_l) and (x_r, y_r). Instead, you threw a rock into the pool which revealed that the point (x_p, y_p) is in the pool (or on the boundary).

Find a way to walk from the start to the end point along a sequence of line segments, so that you never cross the swimming pool.

Input

The first line contains two integers x_s and y_s ($-10^4 \leq x_s, y_s \leq 10^4$).

The second line contains two integers x_t and y_t ($-10^4 \leq x_t, y_t \leq 10^4$).

The third line contains two integers x_p and y_p ($-10^4 \leq x_p, y_p \leq 10^4$).

The problem is not adaptive, i.e. for every test case there exist four integers x_l, y_l, x_r, y_r ($-10^4 \leq x_l < x_r \leq 10^4$, $-10^4 \leq y_l < y_r \leq 10^4$) that constitute a swimming pool. The start and end points are always strictly outside the swimming pool, and the point (x_p, y_p) is inside (or on the border). The start and end points are always distinct.

Output

First, print one integer N ($0 \leq N \leq 10$), the number of points in between the start and end point that you want to visit. Then, print N lines, the ith containing two integers x_i, y_i. These coordinates must satisfy $-10^9 \leq x_i, y_i \leq 10^9$. Note that these are not the same bounds than on the other coordinates.

This means that you will walk along straight line segments between $(x_s, y_s), (x_1, y_1), \ldots, (x_N, y_N), (x_t, y_t)$ such that none of the line segments touch the

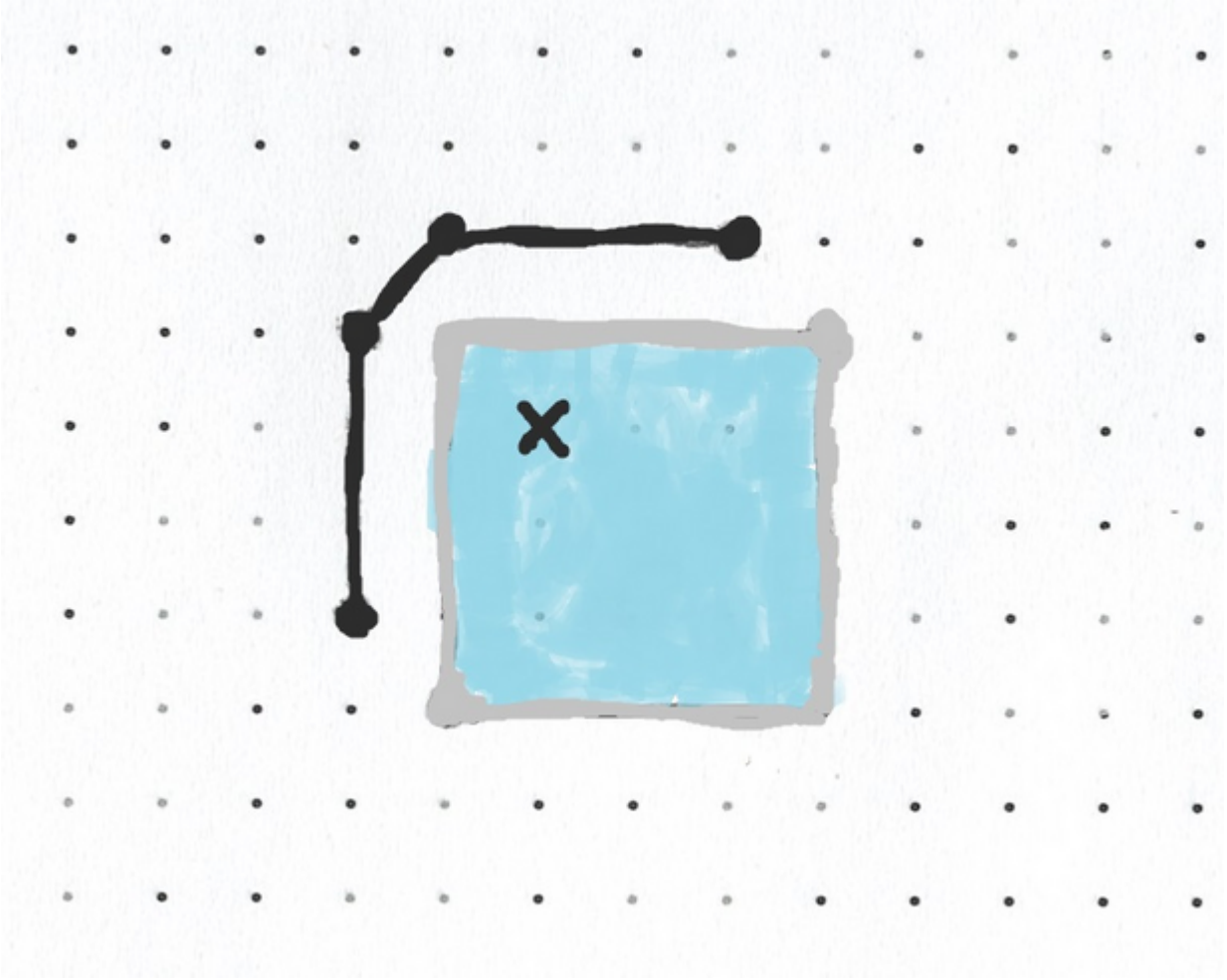
swimming pool. It can be proven that a solution always exists.

Examples

Input	Output
0 0 4 4 2 2	2 0 3 1 4

Note

This picture represents sample 1. The path taken avoids the hidden pool, but based on the information given it could also have intersected it. So the sample solution was quite lucky here.



Problem C. Beth's Cookies

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

This is the story Alph told his schoolmate Beth. "Last vacations, I went exploring that big abandoned house in the woods above the dam lake. I was very proud at that time to be freshly admitted to Tomorrow Programming School, and so I decided to stick to firm and clear rules of exploration and to make a log of my actions."

Inside the house, there were only rooms connected by doors between them. No corridors, no hallways. Whenever I entered a room for the second, third, etc. time, I did it always through the door by which I had left it last time. Whenever I went through a door I wrote down one letter. When I moved from a room to an adjacent room which was either undiscovered yet or which I had discovered later than the current room, I wrote down F, like Forward. When I moved to a room which I had discovered earlier than the current room, I wrote down B, like Backward. By discovery of a room I mean entering the room for the first time.

I started and finished my exploration in the entrance room of the house. When I finally left the house, I had a sequence of letters, in the order I wrote them down during my exploration. Later, to make it more suitable for some future automated processing, I substituted each `F` by an opening bracket and each `B` by a closing bracket. Here is my modified sequence. Do you think it can be used for some unusual programming task?"

It took Beth only seconds to reply. "Surely, it can be used. Insert symbol `*` between each `)(` pair. Insert symbol `1` between each `()` pair. And also, insert pair of symbols `+1` between each `))` pair. It will result in an arithmetic expression, and I will give you as many cookies, as will be the value of the expression, but you have to calculate it first, obviously."

Input

The first input line contains even integer N ($2 \leq N \leq 100$), the length of the sequence of bracket which resulted from Alph's exploration. The second line contains the sequence itself, without any spaces or additional symbols.

Output

Output the number of cookies Alph will receive from Beth.

Examples

Input	Output
10 ((())(()))	5

Problem D. Ammar-utiful Permutations

Time limit	1000 ms
Mem limit	262144 kB
OS	Windows

Turns out Ammar is not that bad at setting problems, as he came up with this problem himself. He wasn't as good as the other judges at coming up with new and fun problem ideas. So instead, he came up with the answer to the problem first, and that's how he proposed this problem.

We define a **permutation** P of **length** N to be called an **Ammar-utiful permutation** if it satisfies the following condition:

- There's **exactly one index** i, $(1 \leq i < N)$ where $P_i + P_{i+1}$ is **odd**.

Given an **integer** N, you must find an **Ammar-utiful permutation** P of length N.

A permutation of length N is an array of size N consisting of N **distinct** integers in the range $[1, N]$. For example, $\{3, 2, 4, 1\}$ is a permutation of length 4, but $\{3, 3, 1, 4\}$ and $\{2, 3, 4, 5\}$ are not.

Note that if there are multiple valid permutations, print any.

Input

The first line of input contains a single integer T $(1 \leq T \leq 1000)$ — the number of testcases.

The only line of each testcase contains a single integer N $(2 \leq N \leq 10^5)$ – the size of the permutation P.

It's guaranteed that the sum of N over all testcases does not exceed 10^5

Output

For each testcase, print in a separate line any permutation of length N that is an **Ammar-utiful permutation**.

Examples

Input	Output
1 7	6 2 4 3 1 7 5

Problem E. Clubbing

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

Students of Tomorrow Programming School engage in different school programming clubs. Each student is a member of some number of clubs. The clubs are supervised by the Principal Club Coach (PCC). His main occupation is to talk to club members to help them organize their activities. PCC has a fixed schedule in the form of a list of students he is going to talk to in the nearest future. He always talks to only one student at a time, he may talk to a student repeatedly at various times. Each talk takes short time interval, which is always the same. There is negligible time between subsequent talks.

Currently, the school director also needs to talk to some students in the presence of PCC, because he needs to start another state supported programming project. He is going to visit PCC's cabinet and spend some uninterrupted time there. In that time, he wants to talk to all members of at least one club. Thus, PCC defined a so-called director interval in his schedule. It is an uninterrupted sequence of his talks to students, in which all members of at least one club appear at least once.

Before he suggests an acceptable director interval, PCC at least wants to know the number of such intervals in his schedule.

Input

The first input line contains one integer N ($1 \leq N \leq 10^5$), the number of student clubs. Next N lines contain the list of club members, each line specifies one club. One club is specified by a string without spaces, in which each member is represented by a single character. All characters in the string are different. The last input line contains the schedule of PCC, in the form of nonempty string with at most 10^5 characters, each character represents one student. In all strings, each character is one of the first 17 lowercase letters in the alphabet("`a`" - "`q`").

Output

Output one integer, the number of director intervals in PCC's schedule.

Examples

Input	Output
2 pid lid lidp	3

Input	Output
2 baf lek affleck	4

Problem F. Expressions

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

Part of the training in arithmetic classes in Tomorrow Programming School consists of quick evaluation of a given arithmetic expression, in the head only. Fortunately, the expressions contain only positive integers, no brackets, and only three operations: addition, subtraction, and multiplication. Thus, the result is guaranteed to be an integer. Moreover, at introductory stages, a student has to evaluate only the parity of the result – is it odd or even?

Unfortunately, the professor presenting the expressions on the blackboard is known to be quite absentminded. He often rewrites various numbers in an expression, even more times, while the students are already calculating. Typically, it changes the value of the whole expression, and the students have to start their calculations all over again.

The students decided to write a program which would help them. The input of the program will be the original expression, and the sequence of subsequent value changes in it. For each change, the program will calculate the value of the modified expression, possibly even without recalculating the whole expression from scratch.

Input

The first input line contains two numbers N and M ($1 \leq N, M \leq 10^5$), the number of integers in the expression, and the number of subsequent value changes in the expression. The second input line contains N token-separated integers $A_1, \ldots, A_N$ ($1 \leq A_i \leq 10^9$). Each token is `+`, `-`, or `*` (plus, minus, star (multiplication)). Each of the next M lines describe one value change in the expression. It contains two space-separated numbers X ($1 \leq X \leq N$) and Y ($1 \leq Y \leq 10^9$), which means that the X-th number in the expression was changed to value Y.

The changes are considered to be subsequent, i.e., when a next change appears on a different position, the previous change in the expression is preserved.

Output

Output $M + 1$ lines. The i-th line contains either "`even`" or "`odd`", depending on whether the input expression evaluates to an even or to an odd value after application of the first $i - 1$ changes. In particular, the first output line indicates whether the expression evaluates to an even or to an odd value before any change was done. The evaluation follows standard arithmetic rule of operations precedence. Multiplication is granted higher precedence than addition and subtraction.

Examples

Input	Output
6 4 11 + 22 * 33 - 44 * 55 * 66 1 2 2 3 4 5 3 5	odd even odd odd odd

Problem G. Fragmentation

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

Not so long ago, a rare type of meteorite, called mesosiderite, fell on the premises of Tomorrow Programming School and was immediately collected. The school was given the honour of organizing the cutting of the meteorite into smaller pieces, which will be exported to laboratories around the world. Swiss Precision Cutting Agency (SPCA) was hired to do the cutting job. SPCA operates various cutting machines. Each machine can cut the meteorite or a piece of the meteorite into a number of smaller pieces, each of which weighs exactly the same. The number of resulting smaller pieces is the characteristic of that particular machine and cannot be changed.

SPCA is quite sure that each of their machines can process any number of meteorite pieces in one day. A piece of meteorite produced by a machine cut cannot be cut again by the same machine in the same day, because of possible cross-contamination and ensuing loss of precision.

Each day, there is exactly one cutting machine available. The order of available machines in successive days is fixed. SPCA created a list of days in the near future and assigned to each day the characteristic of the machine available that day. SPCA calls this list a cutting schedule. The cutting process will take one or more days to complete, before a satisfactory number of meteorite pieces is obtained.

There are a few additional rules governing the cutting and exporting process.

- All laboratories demand that they all receive the same total weight of meteorite pieces.
- No meteorite piece can be shared among laboratories.
- Each piece of the meteorite must go to some laboratory, there should be no leftovers.
- At the end of each day in the cutting process, the weights of all meteorite pieces must be the same, to simplify cutting management.
- The whole cutting process has to be completed in a period of successive days, which should not be interrupted by any day without cutting.

The school has been given the cutting schedule and it is up to them to choose when to perform the actual cutting. A cutting period can be any sequence of consecutive days in the cutting schedule. Clearly, some cutting periods are favourable and some are not. A cutting period is favourable if a cutting process that starts in the first day of the cutting period

would produce meteorite pieces, which satisfy the laboratories' demands at the end of the last day of the cutting period.

The school needs a program that can decide for a cutting period whether it is favourable or not.

Input

The first input line contains integer N ($1 \leq N \leq 10^5$), the number of days in the cutting schedule. The next line contains the cutting schedule in the form of integer sequence $a_1, a_2, \ldots, a_N$ ($1 \leq a_i \leq 10^6$). The value ai represents the characteristic of the machine available on i-th day in the cutting schedule. The next line contains integer Q, ($1 \leq Q \leq 10^5$), the number of following queries. Each of the next Q lines represents one query and it contains three integers s_i, t_i, k ($1 \leq s_i \leq t_i \leq N, 1 \leq k \leq 10^6$), the first day of the cutting period, the last day of the cutting period, and the number of laboratories involved.

Output

For each input query output, on a separate line, `Yes` if the cutting period specified by the query is favourable, otherwise output `No`.

Examples

Input	Output
8 2 3 6 12 4 8 16 4 8 2 4 72 1 8 7 1 4 16 1 4 32 4 4 6 5 5 4 2 5 864 2 5 1296	Yes No Yes No Yes Yes Yes No

Problem H. Golem Coordinated Derby

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

Robotic labs in Tomorrow Programming School produce minirobots in big numbers. To perform various complex tasks, robots often form teams. Before the task begins, a team measures its strength. The robots in the team select one robot among themselves to be the team captain.

Next, the robots arrange themselves in one row behind the captain. Each robot refers to the captain the value of the greatest common divisor of its own height and the height of the neighbour robot standing directly in front of it. That value predicts the strength of the bond between these robots when they perform the task. The captain totals all received values and claims the total to be the strength of the team.

The height of each robot is always expressed in centimeters and it is an integer ranging from 1 to 20. The strength of the team depends on the order of the robots in the row behind the captain. Also note, that a selection of the captain also influences the team strength. Any robot in a team can be selected as its captain.

The robots in a team always tend to maximize the team strength by selecting an appropriate captain and positioning themselves appropriately in the row. However, that is not an easy exercise for the robots, because checking all their possible arrangements is often beyond their computational scope.

Input

The first input line contains one integer N ($2 \leq N \leq 10^5$), the number of robots in the team.

The second line contains N space-separated integers A_i ($1 \leq A_i \leq 20$), the list of heights of all robots in the team.

Output

Output a single integer, the maximum strength of the team specified in the input.

Examples

Input	Output
7 2 3 12 4 6 4 3	22

Problem I. Hamster

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

Hamster Joe is one of the beloved pets of Tomorrow Programming School. The biology department entrusted the school's "Central Robot Endowed with Smashing AI" (CRESAI) with the job of projecting and building a playpen for Joe. This playpen will be situated in a newly designated animal space within the department. This space is a flat horizontal area, tiled with uniformly sized square tiles of unit side length, that come together to create a rectangular grid marked by grooves where tiles meet. The playpen will be bordered by unit-length wall segments, each rising from the groove between two adjacent tiles, ensuring that both ends of every segment align with tile corners.

Joe can move from his current tile to any adjacent tile if the tiles are not separated by a wall segment. Joe cannot jump or crawl over any wall segment and Joe also cannot squeeze himself between two adjacent wall segments, or smash himself through them. Thus, when the locations of the wall segments are chosen appropriately, they form an enclosure, from which Joe would not be able to escape.

Surprisingly, CRESAI was not up to the task. It positioned each wall segment of unit length correctly, in the sense that each segment's base now sits in a groove and its ends coincide with the corners of a tile. However, it seems that most of the wall segments were positioned randomly so that the existence of any enclosure of any shape is not guaranteed.

To fix the problem at least temporarily, biologists are looking for the minimum number of additional wall segments of unit length, which, when installed at appropriately selected unused grooves, would create an enclosure of any shape or size. In the resulting layout, some of the wall segments originally installed by CRESAI may remain useless.

Input

The first line contains one integer M ($1 \leq M \leq 10^5$), the number of wall segments installed by CRESAI. Next, there are M lines, each describing one wall segment installed by CRESAI. A wall segment is described by four integers x_1, y_1, x_2, y_2, where x_1, y_1 are the coordinates of one end of the wall segment and, x_2, y_2 are the coordinates of the other end of the segment. The axes of the system of coordinates are parallel to the grooves and the

coordinates of each grove intersection are integers. For all wall segment coordinates, it holds $-10^9 \leq x_1, y_1, x_2, y_2 \leq 10^9$ and $|x_1 - x_2| + |y_1 - y_2| = 1$.

Output

Output one line with the minimum number of additional wall segments which would create an enclosure of any size or shape, from which Joe would not be able to escape.

Examples

Input	Output
9 0 0 0 1 0 1 1 1 1 1 1 2 1 2 0 2 0 2 -1 2 -1 2 -2 2 -2 2 -2 1 -2 1 -2 0 -2 0 -1 0	1

Note

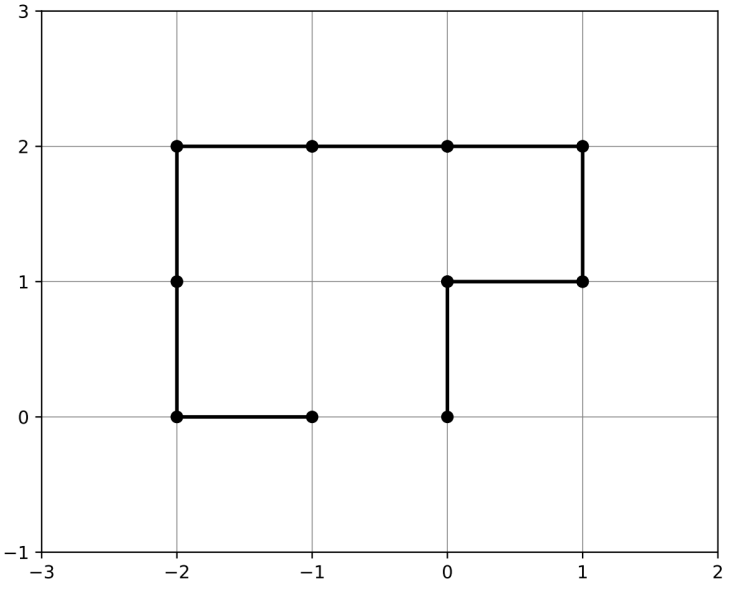


Figure 1: Rendering of the sample input

Problem J. Proglute

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

Baroque opera geeks in Tomorrow Programming School are developing a new string instrument suitable for their innovative performances. The instrument, named programmed lute, or proglute in short, consists of flat circular body, with N pegs situated on the body perimeter, and labeled by integers from 1 to N.

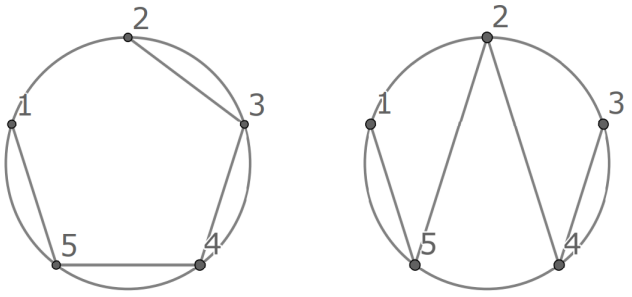
Each string on the instrument is stretched between two different pegs and runs across the proglute's body without crossing other strings. To enhance complex resonance effects, the developers decided to attach two strings to all but two pegs, called principal pegs. Only one string is attached to each principal peg. To support glissando effects, the strings are arranged in such way that it is possible for a musician to touch the string at one principal peg and then slide the finger along all strings on the instrument to the other principal peg. While sliding, musician does not remove the finger from a string, and skips from a string to another one only at their common end peg.

To build the instrument, there are many ways to arrange the strings on the proglute. Different arrangement would result in different musical properties of the instrument. The developers want to know the number of all possible arrangements of strings on the proglute. They introduced the following notions.

- The characteristic of a string is an unordered pair of labels of pegs at the string ends.
- The characteristic of a proglute is the set of all its string characteristics.
- Two strings arrangements on proglute are considered to be different when their corresponding characteristics are different.

Calculate the number of different string arrangements on the proglute.

The figure below shows two possible solutions for a proglute with five pegs:



Input

The input consists of single line with an integer N ($2 \leq N \leq 1000$), the number of the pegs on the proglute perimeter.

Output

Output a single number equal to the number of mutually different arrangements of the strings on the proglute mod $10^9 + 7$.

Examples

Input	Output
5	20

Input	Output
666	61847156