

Problem A. Perfect Alliance

Input file: **standard input**
Output file: **standard output**
Time limit: 0.5 seconds
Memory limit: 512 megabytes

In the AGM world there are N tribes (numbered from 1 to N) and M **directed** roads between these tribes. For each tribe i we know the cost C_i to build a road that leaves this tribe or reaches this tribe. So, the cost to build a directed road from tribe x to tribe y is $C_x + C_y$.

These tribes want to make a *perfect alliance*. They are in a perfect alliance if and only if they can travel from any tribe to any other tribe using the directed roads. Help the tribes by calculating the **minimum** total cost to build some new directed roads, so that they can make a perfect alliance, and tell them what roads they need to build.

Input

The first line of the input contains 2 integers: $1 \leq N \leq 4.000$, the number of tribes; $1 \leq M \leq 20.000$, the number of directed roads between the tribes.

The second line contains the array C of length N , C_i being the cost to build a road that leaves or reaches the tribe i , $1 \leq C_i \leq 10^9$.

Each of the following M lines contains 2 integers: x_i and y_i , representing that there exists a directed road from tribe x_i to tribe y_i , $1 \leq x_i, y_i \leq N$.

Output

The first line of the output should contain the minimum cost to build some new directed roads, so that you can travel from any tribe to any other tribe using the directed roads.

The second line should contain the number of new roads Q .

Each of the following Q lines should contain 2 integers, separated by a space: x_i and y_i , representing a new road from tribe x_i to tribe y_i .

Examples

standard input	standard output
6 6 2 5 3 5 2 7 1 2 1 4 4 5 5 3 5 6 6 4	12 2 3 1 2 5
7 7 2 5 2 5 2 7 8 1 2 2 3 3 4 2 4 4 1 5 6 5 7	23 3 6 3 3 5 7 1

Note

The first example:

We can observe that after we add the roads $(3, 1)$ and $(2, 5)$, we can travel from any tribe to any other tribe using the roads. The cost to add the road $(3, 1)$ is $C_3 + C_1 = 3 + 2 = 5$ and the cost to add the road $(2, 5)$ is $C_2 + C_5 = 5 + 2 = 7$, so the total cost is 12. This is the minimum possible cost.

Problem B. Rainbow Array

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 512 megabytes

At AGM we love colors, as every problem has a unique color. Being passionate about rainbows, you noticed that the spectrum indicator of a rainbow is equal to D .

You are given an array of colors (each color is a non-negative integer value). A sequence of colors is called a *rainbow sequence* if the sum of its colors is divisible by D . Every time you look at the array you want to see a *rainbow sequence* of colors. Because your visibility area is not that large, at some point, you can look at exactly K consecutive positions from the array.

For each position i in the array, it costs you $cost_i$ to change the color from that position to any other color (any other non-negative integer value). You have to answer Q queries. For every query, you are given a position pos from the array and you need to find out the **minimum** total cost to change the array, so that each subsequence of length K (each sequence of K consecutive positions from the array) becomes a *rainbow sequence*, but you can **not** change the color from the position pos .

Input

The first line of the input contains 3 integers: $2 \leq N \leq 500.000$, the length of the array of colors; $2 \leq K \leq \min(100, N)$, your visibility area length; and $1 \leq D \leq 500$, the spectrum indicator of a rainbow.

The second line contains the array col of length N , the array of colors, $1 \leq col_i \leq 10^9$.

The third line contains the array $cost$ of length N , $cost_i$ being the cost of changing the color from position i to any other color, $1 \leq cost_i \leq 10^9$.

The fourth line contains one integer Q , representing the number of queries, $1 \leq Q \leq 500.000$.

Each of the following Q lines contains one integer $1 \leq pos_i \leq N$, the position that can not be changed for this particular query.

Output

The output should contain Q integers, each on a separate line, representing the minimum cost for each query.

Examples

standard input	standard output
7 3 5 5 10 3 15 20 8 2 2 2 3 2 3 3 4 2 5 4	4 9
14 4 7 2 8 10 13 6 8 10 22 25 3 11 13 12 21 1 1 1 1 1 1 1 1 1 1 1 1 1 1 5 8 5 2 7 13	8 9 7 7 8

Note

The first example:

For the first query we can not change the position 5. A solution would be to change the color on the position 1 to 7, and the color on the position 4 to 12. The resulting array will be (7, 10, 3, 12, 20, 8, 2), in which all subsequences of length 3 have the sum of the elements divisible by 5. The cost of these changes is $2 + 2 = 4$. This is the minimum possible cost.

For the second query we can not change the position 4. A solution would be to change the color on the position 7 to 10, the color on the position 2 to 12, and the color on the position 5 to 22. The resulting array will be (5, 12, 3, 15, 22, 8, 10), in which all subsequences of length 3 have the sum of the elements divisible by 5. The cost of these changes is $4 + 2 + 3 = 9$. This is the minimum possible cost.

Problem C. Ohara's Bits

Input file: standard input
Output file: standard output
Time limit: 3.5 seconds
Memory limit: 1024 megabytes

Nix has stumbled upon the legendary quest of Ohara. There are two arrays of numbers, of length N and M respectively. For each number from the first array, Ohara wants to know what is the lowest and highest value of an Ohara match. An Ohara match is defined as follows:

- For each number from the first array, we consider its binary representation **without leading zeros**. This representation, altered by flipping (0 $\rightarrow$ 1 or 1 $\rightarrow$ 0) **at most** one bit, is called an Ohara number.
- Consider the concatenation of the binary representations, **without leading zeroes**, of each number from the second array (from position 1 to position M). The string obtained is called the Ohara string.
- A match appears when there exists a substring $[s_i, s_{i+1}, \dots, s_j]$ in the Ohara string, that is equal to an Ohara number K from the first set.
- If K was obtained without altering the original number from the first array, the value of an Ohara match is 0.
- If K was obtained by altering the original number, by flipping a single bit, then consider the substring from the Ohara string that corresponds to an Ohara match. Now consider the number from the second array, that the flipped bit is part of, in that substring. The value of the Ohara match is equal to the power of 2 corresponding to the flipped bit, in that number (see examples if unclear).

Nix finds this quest rather challenging, so she asks for your help in finding the answer.

Input

The first line of the input contains the numbers N and M , the length of the first and second array of numbers. The second line contains N integers $a_1, a_2, \dots, a_N$ ($1 \leq N \leq 10^5$, $1 \leq a_i < 2^{21}$), the numbers from the first array. The third line contains M integers $b_1, b_2, \dots, b_M$ ($1 \leq b_i \leq 10^9$), the numbers from the second array. It is also guaranteed that the length of the Ohara string is at most 10^5 .

Output

The output will contain N lines. The i^{th} line will contain a pair of integers: the smallest and highest value of an Ohara match for the i^{th} number in the first array. If there is no Ohara match for that number output -1 -1.

Examples

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

Note

In the first example we get the lowest value matching by transforming 101 into 100 and matching on first bit

of 4 from the second array, thus $20 = 1$. The highest value matching is obtained by transforming 101 into 001 and matching on the second bit of 4 from the second array, thus $21 = 2$.

Problem D. Data Integrity

Input file: `standard input`
Output file: `standard output`
Time limit: 2 seconds
Memory limit: 512 megabytes

A data integrity technique is used to ensure that some data is recorded as intended. It is very important to be able to identify that some data was not transmitted correctly so that it can possibly be fixed.

Bob came up with a revolutionary way of storing data using an undirected graph with N nodes. He will be storing data as labels on the edges, such that each label is an integer in $[0, VAL]$. Now, he is curious about the number of distinct data sets that he could possibly place on the graph.

He considers a data set to be valid if and only if for every cycle in the graph, the xor sum of the labels of edges forming that cycle is 0.

Formally, a data set is valid if for every sequence of nodes $a_1, a_2, \dots, a_n$ with $a_1 = a_n$ such that for every integer i in $[1, n - 1]$ there exists an edge b_i connecting a_i and a_{i+1} then:

$label_{b_1} \oplus label_{b_2} \oplus \dots \oplus label_{b_{n-1}} = 0$, where $\oplus$ is the exclusive or operator.

Initially, the graph has M edges, but Bob likes to experiment, so he will modify the graph Q times by either removing an edge or adding a new edge that isn't already in the graph. He would like you to help him find the number of distinct valid data sets that he could have for the initial configuration, as well as after each update.

Input

The first line of the input contains 4 integers N , M , Q , VAL :

$1 \leq N \leq 10^5$, the number of nodes in the graph, $1 \leq M \leq 2 * 10^5$, the number of edges in the initial graph, $0 \leq Q \leq 10^5$, the number of queries and $1 \leq VAL \leq 10^{18}$, the maximum value of a label. It is guaranteed that $VAL = 2^k - 1$ for some integer k .

The next M lines contain a pair of nodes (u, v) , $u \neq v$, representing an edge in the initial graph. It is guaranteed there are no duplicate edges.

The next Q lines contain a pair of nodes (u, v) , $u \neq v$, representing the updates. If the edge is not in the graph, it should be inserted and otherwise it should be removed. **The modifications applied also affect all future updates.**

Output

The output should contain $Q + 1$ integers representing the number of valid data sets for the initial graph as well as after each query, modulo $10^9 + 7$. Two data sets are considered distinct if there exists an edge such that its label's value differs among the two sets.

Examples

standard input	standard output
3 3 2 1 1 2 2 3 3 1 1 3 3 2	4 4 2
7 7 2 65535 1 2 2 3 3 1 1 4 5 6 6 7 7 5 4 1 4 6	496641140 582344008 496641140

Note

In the first example, initially, the 4 possible data sets are $(0, 0, 0)$, $(0, 1, 1)$, $(1, 0, 1)$, $(1, 1, 0)$
After the first update, we are left with edges $1 - 2$ and $2 - 3$, the 4 possible data sets being $(0, 0)$, $(0, 1)$, $(1, 0)$, $(1, 1)$.
After the second update, we are left with just one edge that can be either 0 or 1.

Problem E. Two Gangs

Input file: **standard input**
Output file: **standard output**
Time limit: 0.5 seconds
Memory limit: 512 megabytes

The Federal Intelligence has cracked the supply chains of two major drug cartels in the state of Middleware. The two supply chains are represented by two trees that share the same N nodes. Each node represents a city and appears in each of the supply chains exactly once.

In addition to the exact layout of the two supply chains, the intelligence also uncovered a list with the approximate number of major producers for each supply node. So, for each node in the first supply chain the Feds know a_i , which they assume is the number of major producers among its suppliers. Similarly, they know a list b_i in connection with the second supply chain.

The State Detectives have been tasked to uncover and eliminate the major producers. However, since they are corrupt and receive regular payments from the dealers, they aren't going to make a big fuss out of it: They will instead raid a minimum amount of city hubs such that for each existing city i , at least a_i cities in its subtree within the first supply chain, and at least b_i cities in its subtree within the second supply chain will have been raided. This way, the Feds won't suspect anything.

Input

The first line of the input contains N , the number of cities.

Each of the following $N - 1$ lines contain two numbers x and y , meaning that city y is supplied by city x in the first chain, $1 \leq x, y \leq N$.

Line $N + 1$ contains N numbers: $a_1, a_2 \dots a_N$.

Similarly, the next N lines describe the information for the second supply chain.

$1 \leq N \leq 100$ and $0 \leq a_i, b_i \leq$ their respective subtree sizes.

Output

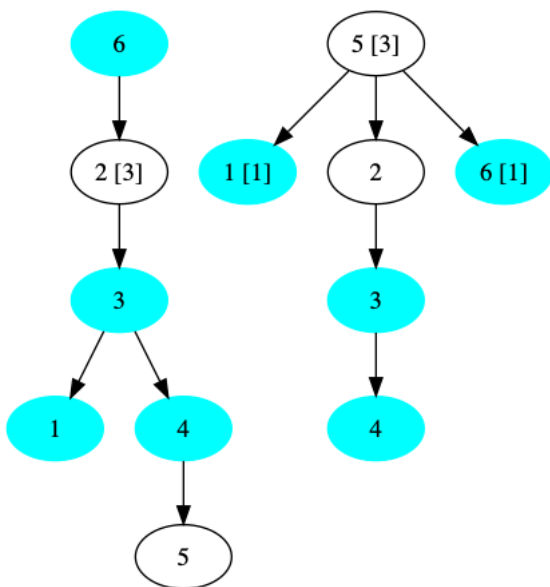
The first line of the output should contain one integer K , representing the minimum number of cities to raid such that all the thresholds in the two supply chains are met.

The second line of the output should contain K integers, representing a list of raided cities such that all the thresholds in the two supply chains are met. If there are multiple solutions, print any of them.

Examples

standard input	standard output
6 3 4 2 3 3 1 6 2 4 5 0 3 0 0 0 0 5 2 2 3 3 4 5 1 5 6 1 0 0 0 3 1	4 1 3 4 6
7 3 4 3 6 4 7 3 2 6 5 6 1 0 0 3 1 0 3 0 7 5 5 1 1 2 2 6 6 3 3 4 2 2 0 0 1 0 0	4 1 4 5 6

Note



Problem F. Japanese parser

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

At the peak of his pandemic-induced boredom, Johnny decided to write his own Japanese parser. He already figured a way to map romaji (Latin transliteration of Japanese characters) to kana (the two Japanese alphabets), but he still needs a way to split his text into romaji.

Given a block of text without spaces, consisting of lowercase Latin characters and basic punctuation marks (., ; ! ? - ()), Johnny needs this text split into romaji (and punctuation marks), separated by a single space.

Romaji are split into two categories: simple and compound. They are listed below. Compound romaji will be treated the same as simple ones.

Romaji	Hiragana	Katakana	Romaji	Hiragana	Katakana
a i u e o	あいうえお	アイウエオ			
ka ki ku ke ko	かきくけこ	カキクケコ	kya kyu kyo	きやきゅきよ	キヤキユキョ
ga gi gu ge go	がぎぐげご	ガギグゲゴ	gya gyu gyo	ぎやぎゅぎよ	ギヤギユギョ
sa shi su se so	さしすせそ	サシスセソ	sha shu sho	しやしゅしよ	シャシュショ
za ji zu ze zo	ざじずぜぞ	ザジズゼゾ	ja ju jo	じやじゅじよ	ジャジュジョ
ta chi tsu te to	たちつてと	タチツテト	cha chu cho	ちゃちゅちょ	チャチュチョ
da di dzu de do	だぢづでど	ダヂヅデド	dya dyu dyo	ぢやぢゅぢよ	ヂャヂュヂョ
na ni nu ne no	なにぬねの	ナニヌネノ	nya nyu nyo	にやにゅによ	ニヤニュニョ
ha hi fu he ho	はひふへほ	ハヒフヘホ	hya hyu hyo	ひやひゅひよ	ヒヤヒュヒョ
ba bi bu be bo	ばびぶべぼ	バビブベボ	bya byu byo	びやびゅびよ	ビヤビュビョ
pa pi pu pe po	ぱぴぷぺぽ	パピプペポ	pya pyu pyo	ぴやぴゅぴよ	ピヤピュピョ
ma mi mu me mo	まみむめも	マミムメモ	mya myu myo	みやみゅみよ	ミヤミュミョ
ya yu yo	やゆよ	ヤユヨ			
ra ri ru re ro	らりるれろ	ラリルレロ	rya ryu ryo	りやりゅりよ	リヤリュリョ
wa wo we wi	わをゑゐ	ワヲヱヰ			
n	ん	ン			

On top of these, Johnny needs to take extra care for some rules. In Japanese, some consonants are "double consonants" and they are marked in a special way. For the purposes of his parser, he decided to mark this by separating the doubled consonant as a lone character. For example "kitte" will be transformed into "ki t te". Double consonants can only appear only at the beginning of a romaji.

In cases where there's ambiguity, Johnny decided that simple romaji take precedence over compound ones (so "nyu" becomes "n yu"), but longer simple romaji take precedence over short ones (so "nu" remains "nu").

To make sure his parser is thoroughly tested, Johnny is using a combination of extracts from articles in Japanese and files he generated randomly, using the rules outlined above.

Input

The input file consists of text up to 100000 lowercase Latin characters and punctuation marks, guaranteed

to lead to a correct solution.

Output

The output should be a sequence of romaji, separated by spaces. Punctuation marks should be treated the same way.

Examples

standard input	standard output
arigatougozaimasu	a ri ga to u go za i ma su
tottemogenkidesu	to t te mo ge n ki de su
nazekorewohonyakushitanodesuka	na ze ko re wo ho n ya ku shi ta no de su ka

Problem G. Complete Journey

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 512 megabytes

You will make a trip in the mountains! There are N peaks (numbered from 1 to N) you want to visit and M undirected trails connecting these peaks; each trail has a **distinct** value of beauty. It is guaranteed that you can reach any peak from any other peak using the trails.

The value of beauty of a path between two peaks is the **maximum** value across all the trails that form this path.

In a **journey** between two peaks you will take a path between them with **minimum** value of beauty, because the other ones are considered to be too dangerous. So, the value of beauty of a journey between two peaks is the **minimum** value of beauty of a path among all the paths between these two peaks.

In your trip you want to make a *complete journey*. You want to visit all the peaks in a fixed order by making multiple journeys, so that the sum of values of beauty of all the journeys is as much as possible.

In other words, you need to find a permutation of the peaks $\{p_1, p_2, \dots, p_N\}$, so that the sum of values of beauty of the journeys (p_1, p_2) , (p_2, p_3) , ..., (p_{N-1}, p_N) is the **maximum** among all the permutations.

Input

The first line of the input contains 2 integers: $1 \leq N \leq 10^5$, the number of peaks; $1 \leq M \leq 2 * 10^5$, the number of trails.

Each of the following M lines contains 3 integers: x_i, y_i, c_i , representing a bi-directional trail between the peaks x_i and y_i having the value of beauty c_i , $1 \leq x_i, y_i \leq N$, $1 \leq c_i \leq 10^9$. All values of beauty are distinct.

Output

The first line of the output will contain $CMax$, the maximum total value of beauty of a permutation.

The second line will contain a valid permutation of length N , having the total value of beauty $CMax$.

Examples

standard input	standard output
3 2 1 2 1 1 3 2	4 1 3 2
10 17 2 10 1 10 3 2 10 6 3 2 1 4 2 3 5 3 5 6 1 4 7 4 5 8 5 6 9 4 8 10 1 10 11 8 7 12 1 9 13 9 3 14 6 7 15 4 9 16 9 5 17	90 2 9 5 6 4 3 8 1 7 10

Note

In the first example, the value of beauty of the journey (1,3) is 2, and the value of beauty of the journey (3,2) is 2, hence the sum is 4. This is the maximum possible sum.

Problem H. KMP

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 512 megabytes

Peter has a sequence S of numbers of length N and he wonders how many *kompakt* subsequences are there in his sequence.

A subsequence of S is a non-empty sequence $S_{p_1}, S_{p_2}, \dots, S_{p_K}$ with $1 \leq p_1 < p_2 < \dots < p_K \leq N$ for some $1 \leq K \leq N$. Two subsequences $S_{p_1}, S_{p_2}, \dots, S_{p_K}$ and $S_{q_1}, S_{q_2}, \dots, S_{q_L}$ are different if the sequences p and q are different (either $K \neq L$, or there exists $1 \leq i \leq K$ such that $p_i \neq q_i$).

A sequence is *kompakt* if it consists only of different values and it contains every integer value between the minimum number in the sequence and the maximum number in the sequence. For example, sequences $[1, 2]$ and $[7, 9, 8]$ are *kompakt*, but sequences $[1, 3]$ and $[7, 9, 6]$ are not.

Help Peter count the number of *kompakt* subsequences in S .

Input

The first line of the input contains $1 \leq N \leq 10^5$, the length of the sequence.

The second line of the input contains N integers, $S_1, S_2, \dots, S_N$, representing Peter's sequence.

$1 \leq S_i \leq 10^5$ for all $1 \leq i \leq N$.

Input

The first line of the input contains $1 \leq N \leq 10^5$, the length of the sequence.

The second line of the input contains N integers, $S_1, S_2, \dots, S_N$, representing Peter's sequence.

$1 \leq S_i \leq 10^5$ for all $1 \leq i \leq N$.

Output

The only line in the output should contain an integer representing the number of *kompakt* sequences in S . Since the number might be too big, print the answer modulo $10^9 + 7$.

Examples

standard input	standard output
6 15 16 18 15 8 19	9
8 7 1 9 11 2 10 8 10	26

Note

The *kompakt* subsequences in the first example are: $[15], [16], [18], [15], [8], [19], [15, 16], [16, 15], [18, 19]$

Problem I. Orchards

Input file: **standard input**
Output file: **standard output**
Time limit: 0.5 seconds
Memory limit: 512 megabytes

The year is 2121. The first humans have reached the planet AGMinia and established a small settlement. They now want to build a number of orchards on a carefully chosen piece of land. However, due to the special climate on the newly discovered planet, they must follow a number of rules as described below.

The land is represented by a two dimensional array of N rows and M columns. Each cell of the land has its own height.

The K non-overlapping orchards they want to build are represented by rectangles within the main piece of land. Two orchards g_1 and g_2 are considered adjacent if there are any two cells c_1 and c_2 such that c_1 is within g_1 and c_2 is within g_2 and c_1 and c_2 are adjacent vertically, horizontally or diagonally.

All the cells within an orchard must be leveled, i.e. have the same height. In order to achieve this, the humans can use advanced technology to increment or decrease the height of a cell by 1. Such an operation can be performed an unlimited number of times on a given cell provided that the height doesn't go below 1. The cost of performing the operation once is 1.

Moreover, due to the special conditions on the planet, any two adjacent orchards must have different heights. And because it is easier to control the weather at the same height, there must be at most two distinct heights among all orchards after they are leveled.

Help the humans improve the new settlement and compute the minimum cost they will need in order to achieve this goal.

Input

The first line of the input contains 3 integers: N, M, K (the number of rows and columns of the land, respectively, and the number of orchards).

Each of the following K lines contains 4 integers x_1, y_1, x_2, y_2 . (x_1, y_1) represents the upper-left corner of the i^{th} orchard and (x_2, y_2) represents the bottom-right corner of the same orchard.

The next N lines contain M integers each, describing the heights of the land.

$1 \leq N, M \leq 1000$ and $1 \leq K \leq 500$.

$1 \leq x_{i1} \leq x_{i2} \leq N, 1 \leq y_{i1} \leq y_{i2} \leq M$, for all $1 \leq i \leq K$.

$0 \leq \text{height of any cell} \leq 100$.

$1 \leq \text{height of any cell within an orchard} \leq 100$.

It is guaranteed that no two orchards will overlap.

Output

The output should contain one integer, representing the minimum possible cost needed to complete the task, or -1 if it is not possible.

Examples

standard input	standard output
3 5 3 1 1 2 3 1 5 2 5 3 1 3 3 1 1 1 0 1 1 1 1 0 2 1 2 2 0 0	2
1 5 3 1 1 1 1 1 3 1 3 1 5 1 5 7 2 7 4 7	0
1 5 3 1 1 1 1 1 3 1 3 1 5 1 5 7 2 8 4 9	1
2 2 4 1 1 1 1 1 2 1 2 2 1 2 1 2 2 2 2 1 1 1 1	-1

Note

In the first example, there are two adjacent orchards (the first and the last one). The first orchard is already leveled. We can level the third one by increasing the height of the first cell of the third row with one. For the second orchard, we can either increase the shorter cell or decrease the other one. Two operations is the best we can do here, thus the answer is 2.

For the second example, all the orchards are already leveled as they each contain only one cell and none of them is adjacent to another one. All the criteria is satisfied without having to perform any operation, hence the cost is 0.

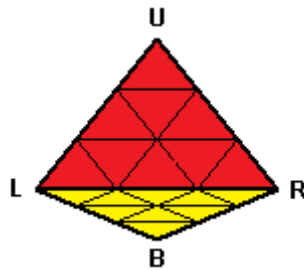
In the third example, the orchards consist of only one cell each, this time having the heights 7, 8 and 9 respectively. We must have at most two distinct heights, so we have to apply one operation here, for example decrease the third one to 8.

In the last example it is impossible to complete the task without breaking any of the rules.

Problem J. Pyra, Pyra, Pyraminx!

Input file: **standard input**
Output file: **standard output**
Time limit: 10 seconds
Memory limit: 512 megabytes

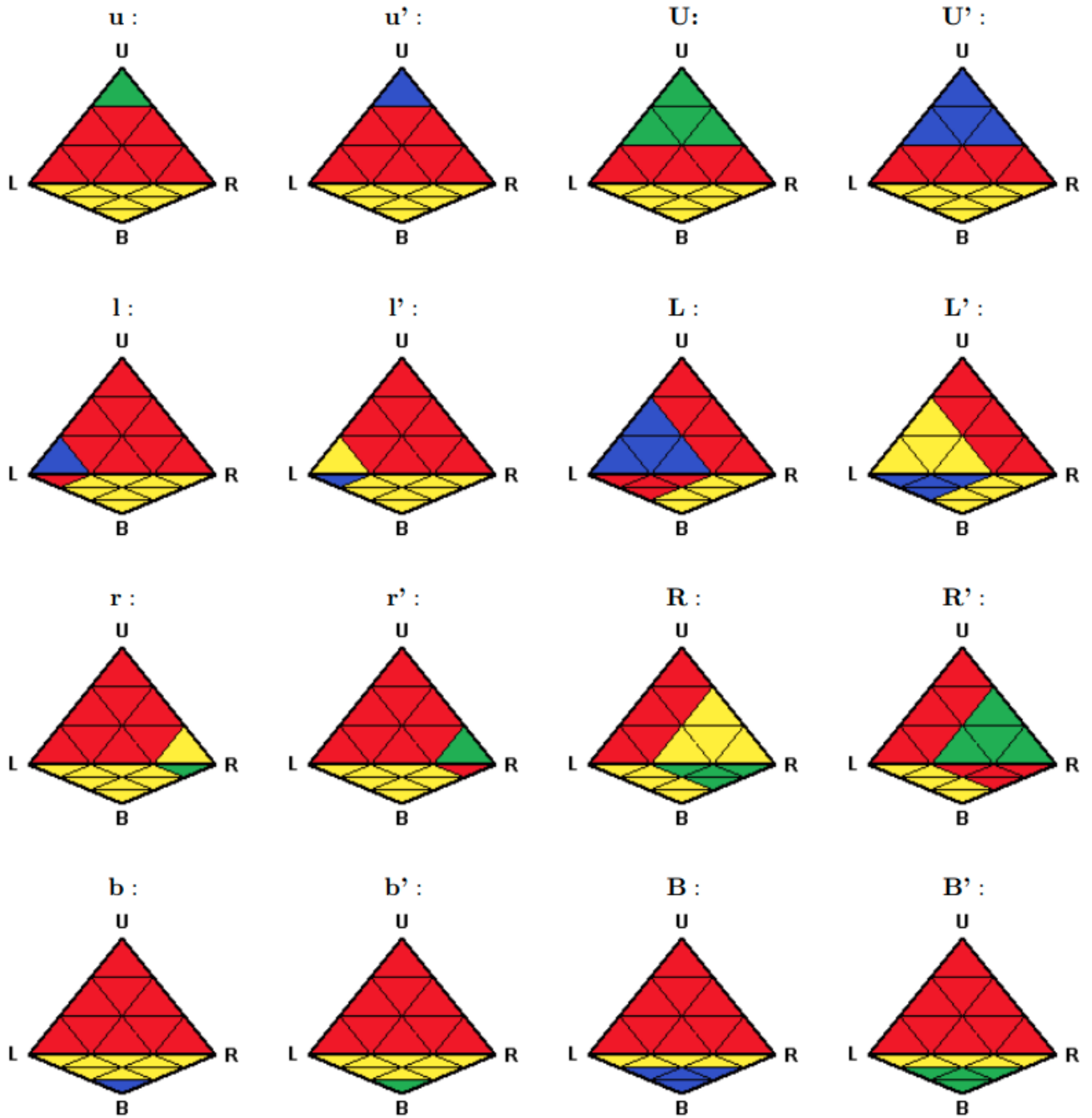
Binic, the speedcuber, has just found a new puzzle that is challenging him: the Pyraminx. This puzzle, contrary to the Rubik's Cube, has the form of a regular tetrahedron (pyramid with triangular base, that has all edges of equal length).



Consider the pyramid as looking directly to a face, then the four tips of the pyramid can be noted with letters **U**(p), **L**(eft), **R**(ight) and **B**(ottom), as in the image to the right. Then, let's say the front face is ULR, the left face is UBL, the right face is URB and the bottom face is BRL.

The pieces of the puzzle can be moved 120° clockwise or counter-clockwise around the axis denoted by a tip of the pyramid and its projection on the opposite face. A rotation of a tip in clockwise direction (when looking straight to that tip) is represented by the lowercase letter of the notation of that tip. A rotation of a tip and the row under it (2 rows of the Pyraminx) in clockwise direction (when looking straight to that tip) is represented by the uppercase letter of the notation of that tip. If an apostrophe (') is added after the letter, the rotation is performed counter-clockwise.

The valid moves for a Pyraminx, starting from the solved state and assuming that the front face is red, the left face is blue, the right face is green, and the bottom face is yellow, are:



Binic is not able to solve this puzzle, but asks you for help. Find a sequence of **at most 100** moves that transforms the Pyraminx in the solved state.

Input

The first line of the input contains a number $1 \leq T \leq 20$, the number of tests.

Each test is described in 12 lines, the first 3 representing the front face (where the upper tip is U), next 3 representing the left face (where the upper tip is U), next 3 representing the right face (where the upper tip is U), and the last 3 representing the bottom face (where the upper tip is B). All faces are described as seen if directly looking at them, knowing that the upper tip is the given one (regarding the initial rotation of the Pyraminx).

The descriptions of each face if formed of 3 lines, containing 1, 3, respectively 5 characters, representing the colours of the triangles on that face (R - red, B - blue, G - green, Y - yellow).

The configurations given as input are always solvable.

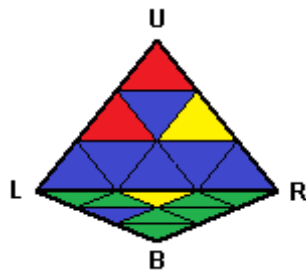
Output

The output should contain the answer for the T tests, for each a sequence of moves that transforms the Pyraminx in the solved state. The sequence of moves is formatted in the following way: on the first line, an integer S , representing the number of moves, followed by S lines, each containing the encoding of a single move.

Example

standard input	standard output
2	14
R	b
RB	R
BY	R'
BBBBB	U'
Y	L
BRG	b
RRGRR	U'
B	L
RYR	U
YYYYY	u
G	L'
GGB	U
GGYGG	L'
B	b
GBG	6
BBBBB	U'
R	L
BRB	U'
RRRRR	L
Y	U'
RYR	L
YYYYY	
G	
GGY	
GGYGG	

Note



The configuration presented in the example's first test case would look the following, before solving:

Problem K. Squares City

Input file: `standard input`
Output file: `standard output`
Time limit: 1 second
Memory limit: 512 megabytes

After a long journey around the world, you found a mythic place called *Squares City*. This city is special, and a little bit strange, since all the buildings have the shape of squares, and even the city itself can be seen as a squared matrix. What is even more interesting is that, after a short walk around the city, you also realised that all the buildings and the city have lengths equal to powers of two. However, even if you found all this information captivating, your mathematical skills awakened, and now you are worried if there would be a better way to build such a city. After talking to the mayor, you found out the rules that their city must respect and you were also given the chance to come up with a new idea that will be further presented to the city council.

The rules are as follows:

1. The city must have the shape of a square of length N , where N is given to be of form 2^K .
2. The buildings that you build must have the shape of squares, and their lengths must be of form 2^K . You are free to choose which length each building should have.
3. You are allowed to build only above the secondary diagonal of the city, since the rest of the city will be used for recreation areas for citizens.
4. The entire area where you are allowed to build must be used (i.e. every single cell above the secondary diagonal must have a building on it).
5. The number of buildings that you build must be minimized.

Given these rules, the city council is going to ask you Q questions of form N, X, Y , i.e. what is the length of the building that would be built over the cell (X, Y) in your optimal representation of the city, if the city has length N ? In case that there are several answers to the question, you should answer with -1 .

Input

The first line of the input contains $1 \leq Q \leq 10^5$, the number of queries.

The following Q lines will each contain integers (N, X, Y) signifying a question from the city council ($2 \leq N \leq 2 * 10^{18}$, N of form 2^K , $1 \leq X \leq N - 1$, $1 \leq Y \leq N - X$).

Output

The output should contain Q lines, each of them containing the answer to the question, or -1 if there are several answers to the question.

Example

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

Note

In the given examples, the length of the city is 4, therefore the area where you are allowed to build is represented by the following matrix:

```
1110
1100
1000
0000
```

The only optimal answer is to use a building of length 2 and two buildings of length 1 as follows:

```
2210
2200
1000
0000
```

Problem L. Yet another roads problem

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 512 megabytes

Following a narrative plot that you have never encountered in any other programming contest before, you are the Ministry of Transport in country Foo. This realm's historical context led to all of the cities being secluded from each other. Although the country is doing amazing as it is, we would not have a problem statement without changing something.

Therefore, I have now decided that the country is doing horrible, and the reason why is that you cannot move from a city to another. The solution is simple: just build roads.

Although you could simply build roads between all cities and call it a day, there is a plot twist - the country is poorTM. Therefore the number of roads built must be minimized, while also ensuring that citizens can travel from any city to another.

In addition, building a road between two cities will contribute to Foo's economic growth. For inexplicable reasons, this growth is given by the greatest common divisor of the two cities' number of residents.

Given the number of cities, N , and the number of residents of every single one of them, x_i ($1 \leq i \leq N$), it is asked of you to build a minimum amount of roads such that you can travel from any city to another, while also maximizing the total economic growth.

Input

The first line consists of an integer N ($1 \leq N \leq 10^5$), representing the number of cities. The second line will contain N other integers x_i ($1 \leq x_i \leq 10^6$), where x_i denotes the number of residents of city i .

Output

It is asked of you to print an integer which represents the total economic growth of country Foo after building all of the roads. Bear in mind that it must be maximized.

Example

standard input	standard output
7 42 21 5 6 7 3 9	41

Note

A road can be built between the first two cities, generating an economic growth of 21. Connecting the first city to the 4th and 5th ones will further increase the total growth by 6, respectively 7. Moreover, building a road between the last two cities will contribute to the growth by a value of 3. To make all cities reachable from one another, we now build a road between the first and last city, growing the economy by 3, and from the third to whichever city, increasing the total growth by 1.

Therefore we get a total economic growth of $21 + 7 + 6 + 3 + 3 + 1 = 41$.