Problem A. Company Merging

Time limit: 1 second
Memory limit: 512 megabytes

A conglomerate consists of n companies. To make managing easier, their owners have decided to merge all companies into one. By law, it is only possible to merge two companies, so the owners plan to select two companies, merge them into one, and continue doing so until there is only one company left.

But anti-monopoly service forbids to merge companies if they suspect unfriendly absorption. The criterion they use is the difference in maximum salaries between two companies. Merging is allowed only if the maximum salaries are equal.

To fulfill the anti-monopoly requirements, the owners can change salaries in their companies before merging. But the labor union insists on two conditions: it is only allowed to increase salaries, moreover all the employees in one company must get the same increase.

Sure enough, the owners want to minimize the total increase of all salaries in all companies. Help them find the minimal possible increase that will allow them to merge companies into one.

Input

The first line contains a single integer n — the number of companies in the conglomerate ($1 \le n \le 2 \cdot 10^5$). Each of the next n lines describes a company.

A company description start with an integer m_i — the number of its employees ($1 \le m_i \le 2 \cdot 10^5$). Then m_i integers follow: the salaries of the employees. All salaries are positive and do not exceed 10^9.

The total number of employees in all companies does not exceed $2 \cdot 10^5$.

Output

Output a single integer — the minimal total increase of all employees that allows to merge all companies.

Example

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

Note

One of the optimal merging strategies is the following. First increase all salaries in the second company by 2, and merge the first and the second companies. Now the conglomerate consists of two companies with salaries $[4, 3, 4, 3]$ and $[1, 1, 1]$. To merge them, increase the salaries in the second of those by 3. The total increase is $2 + 2 + 3 + 3 + 3 = 13$.

Problem B. LaTeX Expert

Time limit:	1 second
Memory limit:	512 megabytes

Regina is finishing her graduation work. She asks Grigory who is LaTeX expert to help her with layout.

Making layout is easy for Grigory, but he has some problems with bibliography rendering. Due to graduation work design requirements, references in bibliography must be placed in the same order as they are placed in the text of the work. Each reference has exactly one occurrence in Regina's work.

For placing reference on some source Grigory uses `\cite{<reference>}` construction, where `<reference>` is the name of the reference. An example of a text that contains three references is shown below.

```
The most famous characters of Pushkin's works are Onegin \cite{onegin},
Dubrovsky \cite{dubrovsky} and Tsar Saltan \cite{saltan}.
```

To make the bibliography, Grigory needs to place the following construction after the text of the work:

```
\begin{thebibliography}{99}
\bibitem{onegin} A.S.Pushkin. Eugene Onegin. 1831.
\bibitem{dubrovsky} A.S.Pushkin. Dubrovsky. 1841.
\bibitem{saltan} A.S.Pushkin. The Tale of Tsar Saltan. 1832.
\end{thebibliography}
```

Here `99` is the maximal number of references and reference description is written after `\bibitem{<reference>}` construction.

Text is quite large and Grigory is quite lazy, so he doesn't want to check the order of references. Therefore he asks you to write a program that can analyze the text and the bibliography. If the bibliography is incorrect, your program must print the correct new one.

Input

Input contains the text of the work that consists of lowercase and uppercase English letters, ends of lines, spaces, «.», «,», «'» (ASCII code 39) characters and `\cite{<reference>}` constructions. Here `<reference>` is a nonempty string of lowercase English letters that has the length at most 100. ASCII code of «\» character is 92.

Each `\cite{<reference>}` construction starts at a new line or after a space character. The text of the work can contain empty lines.

The following line after the text is equal to `\begin{thebibliography}{99}`. Each of the following lines contains a description of a source in the described format. Description consists of the same characters as the text of work. Length of each description isn't greater than 100 characters. The following line after the bibliography is `\end{thebibliography}`.

It's guaranteed that:

- The total number of references is not greater than 99.
- There is at least one reference in the text.
- If the text contains some reference, the bibliography contains that reference too.
- The number of `\cite{<reference>}` constructions in the text is the same as the number of `\bibitem{<reference>}` constructions in the bibliography.
- The references in the text are different.
- The total number of lines in the input isn't greater than 10^5.
- The sum of lengths of the lines in the text isn't greater than 10^5.
- Input doesn't contain two consecutive space characters. The first and the last characters of each line aren't spaces.

Output

Print `Correct` if the bibliography from the input is correct and `Incorrect` otherwise.

In the second case print the correct bibliography in the required format then. The source descriptions must be the same as defined in input.

Examples

standard input
The most famous characters of Pushkin's works are Onegin \cite{onegin}, Dubrovsky \cite{dubrovsky} and Tsar Saltan \cite{saltan}. \begin{thebibliography}{99} \bibitem{saltan} A.S.Pushkin. The Tale of Tsar Saltan. 1832. \bibitem{onegin} A.S.Pushkin. Eugene Onegin. 1831. \bibitem{dubrovsky} A.S.Pushkin. Dubrovsky. 1841. \end{thebibliography}
standard output
Incorrect \begin{thebibliography}{99} \bibitem{onegin} A.S.Pushkin. Eugene Onegin. 1831. \bibitem{dubrovsky} A.S.Pushkin. Dubrovsky. 1841. \bibitem{saltan} A.S.Pushkin. The Tale of Tsar Saltan. 1832. \end{thebibliography}

standard input
The most famous characters of Pushkin's works are Onegin \cite{onegin}, Dubrovsky \cite{dubrovsky} and Tsar Saltan \cite{saltan}. \begin{thebibliography}{99} \bibitem{onegin} A.S.Pushkin. Eugene Onegin. 1831. \bibitem{dubrovsky} A.S.Pushkin. Dubrovsky. 1841. \bibitem{saltan} A.S.Pushkin. The Tale of Tsar Saltan. 1832. \end{thebibliography}
standard output
Correct

Problem C. New Year Presents

Time limit: 2 seconds
Memory limit: 512 megabytes

Santa has prepared boxes with presents for n kids, one box for each kid. There are m kinds of presents: balloons, sweets, chocolate bars, toy cars... A child would be disappointed to receive two presents of the same kind, so all kinds of presents in one box are distinct.

Having packed all the presents, Santa realized that different boxes can contain different number of presents. It would be unfair to the children, so he decided to move some presents between boxes, and make their sizes similar. After all movements, the difference between the maximal and the minimal number of presents in a box must be as small as possible. All presents in each box should still be distinct. Santa wants to finish the job as fast as possible, so he wants to minimize the number of movements required to complete the task.

Given the sets of presents in each box, find the shortest sequence of movements of presents between boxes that minimizes the difference of sizes of the smallest and the largest box, and keeps all presents in each box distinct.

Input

The first line of input contains two integers n, m ($1 \le n, m \le 100\ 000$), the number of boxes and the number of kinds of the presents. Denote presents with integers from 1 to m.

Each of the following n lines contains the description of one box. It begins with an integer s_i ($s_i \ge 0$), the number of presents in the box, s_i distinct integers between 1 and m follow, denoting the kinds of presents in that box.

The total number of presents in all boxes does not exceed 500 000.

Output

Print one integer k at the first line of output, the number of movements in the shortest sequence that makes the sizes of the boxes differ by at most one. Then print k lines that describe movements in the same order in which they should be performed. Each movement is described by three integers $from_i$, to_i, $kind_i$. It means that the present of kind $kind_i$ is moved from the box with number $from_i$ to the box with number to_i. Boxes are numbered from one in the order they are given in the input.

At the moment when the movement is performed the present with kind $kind_i$ must be present in the box with number $from_i$. After performing all moves each box must not contain two presents of the same kind.

If there are several optimal solutions, output any of them.

Example

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

Problem D. Similar Arrays

Time limit: 1 second
Memory limit: 512 megabytes

Vasya had an array of n integers, each element of the array was from 1 to n. He chose m pairs of different positions and wrote them down to a sheet of paper. Then Vasya compared the elements at these positions, and wrote down the results of the comparisons to another sheet of paper. For each pair he wrote either "greater", "less", or "equal".

After several years, he has found the first sheet of paper, but he couldn't find the second one. Also he doesn't remember the array he had. In particular, he doesn't remember if the array had equal elements. He has told this sad story to his informatics teacher Dr Helen.

She told him that it could be the case that even if Vasya finds his second sheet, he would still not be able to find out whether the array had two equal elements.

Now Vasya wants to find two arrays of integers, each of length n. All elements of the first array must be distinct, and there must be two equal elements in the second array. For each pair of positions Vasya wrote at the first sheet of paper, the result of the comparison must be the same for the corresponding elements of the first array, and the corresponding elements of the second array.

Help Vasya find two such arrays of length n, or find out that there are no such arrays for his sets of pairs.

Input

The first line of input contains two integers n, m — the number of elements in the array and number of comparisons made by Vasya ($1 \le n \le 100\,000$, $0 \le m \le 100\,000$).

Each of the following m lines contains two integers a_i, b_i — the positions of the i-th comparison ($1 \le a_i, b_i \le n$; $a_i \ne b_i$). It's guaranteed that any unordered pair is given in the input at most once.

Output

The first line of output must contain "YES" if there exist two arrays, such that the results of comparisons would be the same, and all numbers in the first one are distinct, and the second one contains two equal numbers. Otherwise it must contain "NO".

If the arrays exist, the second line must contain the array of distinct integers, the third line must contain the array, that contains at least one pair of equal elements. Elements of the arrays must be integers from 1 to n.

Examples

standard input	standard output
1 0	NO
3 1 1 2	YES 1 3 2 1 3 1
4 3 1 2 1 3 2 4	YES 1 3 4 2 1 3 4 1

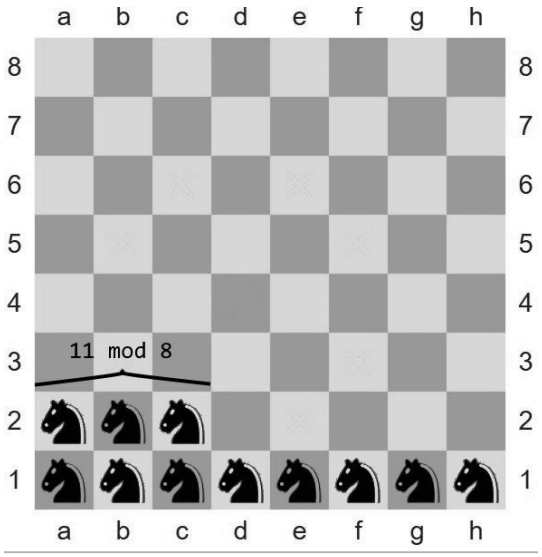
Problem E. Horseback Riding

Time limit:	1 second
Memory limit:	512 megabytes

It is well known that chess is a sport. But Dima, who is into horseback riding, considers it boring, and not a sport at all. His friend Sasha has decided to show him how complex and interesting chess can be.

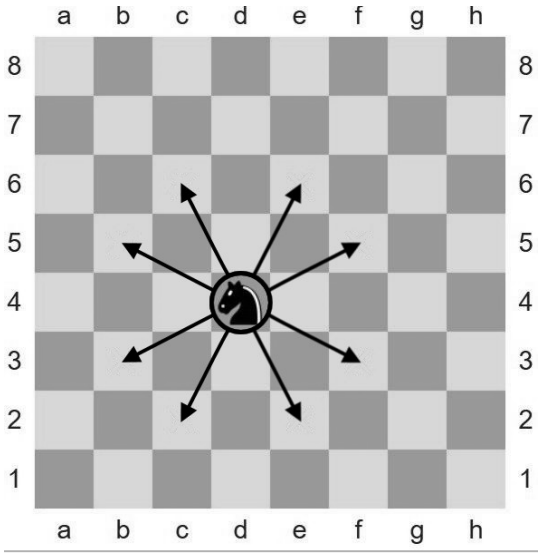
Sasha has placed k knights on a 8×8 chessboard. These knights are on a promenade and want to return to their stables. Unfortunately, they don't remember the way home.

Let us say that the knights are in their stables if the following conditions are satisfied: some (specifically, k div 8) lower rows are filled with knights, and the next row only contains knights in its leftmost cells (if $k \bmod 8 \neq 0$, then $k \bmod 8$ knights take leftmost positions in this row).



There are the desired cells for $k = 11$ knights, one per knight.

This is a chess problem, so the knights move as they do in chess: exactly two cells in one direction, and then exactly one cell in another direction.



Possible moves of a knight

Knight make moves one by one, in any order. In a single moment at most one knight can be located in any cell.

Dima spent two days solving this problem, and now asks you to find the sequence of knights moves, so that no two knight ever occupied the same cell, and all the knight ended up in the stables. You don't have to minimize the number of moves, but you are limited to no more than 1500 moves in total.

Input

The first line of input contains an integer k — the number of knights at the chessboard ($1 \le k \le 64$). The following k lines contains the positions of the knigths. Each position is formatted as `xy`, where `x` is the column letter, and `y` is the row digit.

Columns are numbered from «`a`» to «`h`» left-to-right, rows are numbered with digits from 1 to 8 bottom-to-top.

Output

The first line of output must contain the number of moves you need to get knights into the stables. Next, output these moves one per line, in the order they are made. Format your moves as `xy-zt`, whehe `x` and `z` are column letters, and `y` and `t` are row digits.

Remember that you don't have to minimize the number of moves, but you have to fit into the limit of 1500.

Examples

standard input	standard output
2 a5 f1	4 a5-b3 b3-a1 f1-d2 d2-b1
3 a1 b3 c2	5 b3-d2 d2-b1 c2-b4 b4-d3 d3-c1

Problem F. How to Learn You Score

Time limit: 1 second
Memory limit: 512 megabytes

This is an interactive problem.

Petya participates in such contest that the participants only learn results of judging their submissions after the end of the contest.

There are n problems numbered from 1 to n in the contest. For each problem he has made a submission. After the contest he was very eager to learn his points and his place. But the results would only be published at the closing ceremony, which is a few hours away.

Petya decided to find out his result before that: to get his points for each problem from the jury, he would ask weird questions. Let Petya's points be $c_1, c_2, \ldots, c_n$ for problems $1, 2, \ldots, n$, respectively. Petya doesn't know these numbers and wants to find them out.

Petya chooses three distinct numbers of the problems and asks the jury to tell him the sum of the minimum and the maximum points he received for these three problems. Fortunately, the jury answers such queries. Formally, if he queries about i, j, k, he gets $min(c_i, c_j, c_k) + max(c_i, c_j, c_k)$.

Petya still doesn't know how to find out all his points. Help him find out $c_1, c_2, \ldots, c_n$, by making no more than $4n$ queries to the jury.

Interaction Protocol

First, your program must read a single integer n — the number of the problems in the contest ($5 \le n \le 1000$).

Next, your program can make queries about the sum of minimum and maximum points among problems i, j, k. In order to do that, output a separate line «? i j k» ($1 \le i, j, k \le n$, $i \ne j$, $j \ne k$, $i \ne k$). As the answer to this query, you will get a single integer $min(c_i, c_j, c_k) + max(c_i, c_j, c_k)$ in the input.

If your program makes more than $4n$ queries or asks an invalid query, you get «`Wrong Answer`».

After finding Petya's points output «! c_1 c_2 ... c_n» — the points for each problem.

It is guaranteed that Petya's points for each problem are integer and satisfy $0 \le c_i \le 10^9$ for all $1 \le i \le n$. Also, all numbers c_i are fixed in advance and won't change during the testing of your program.

Example

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

Note

Print new line character after each line. After each action your program must flush standard output. If you use "`writeln`" in Pascal, "`cout << ... << endl`" in C++, "`System.out.println`" in Java, "`print`" in Python, "`Console.WriteLine`" in C#, standard output is automatically flushed, no additional action is required. If you are using other way to output data, you should flush standard output. Note that even if you flush output, you absolutely must print new line character.

Explanation

In this sample, there are 5 problems, and Petya got 1, 0, 2, 1, 3 points for the problems. If he queries about problems 1, 2, 3, his minimum points among them is 0 (for the 2-nd problem), and his maximum is 2 (for the 3-rd problem). Thus, the jury will answer $2 = 0 + 2$ to that query.

Problem G. Combostone

Time limit:	2 seconds
Memory limit:	512 megabytes

One big company is developing a computer game named "Combostone", that should blast the market. The rules of the game are quite complicated, so implementation of the server engine, which should adhere to the rules, was outsourced. You are asked to implement such server engine.

The players of the game are able to summon creatures to the arena, and then they can improve these creatures. Every creature has two parameters — an integer value of its attack a and an integer value of its remaining health points h. Let us denote parameters of a creature as (a, h). There are no creatures in the arena at the beginning of the game.

Players are able to use the following spells:

- *Summon Creature*: Summon new creature with parameters $(1, 1)$. If k creatures have already been summoned, the new creature gets an id $k + 1$.
- *Blessed Champion*: Double the attack of the chosen creature. If the creature have had parameters (a, h) before this spell, after the spell the creature has parameters $(2a, h)$.
- *Divine Spirit*: Double the health points of the chosen creature. If the creature have had parameters (a, h) before this spell, after the spell the creature has parameters $(a, 2h)$.
- *Molten Reflection*: Summon the new creature, which has the same parameters as the chosen creature. If k creatures have already been summoned in the game, the new creature gets an id $k + 1$.
- *Attack!*: Make a fight between two different creatures. In the fight both creatures at the same time make one hit to the enemy. This hit decreases opponent's health points by the attack value of the creature. In other words, after the fight between creatures with parameters (a_1, h_1) and (a_2, h_2) their parameters become $(a_1, h_1 - a_2)$ and $(a_2, h_2 - a_1)$, respectively. If after the fight the creature is left with 0 or less health points, it dies and can't participate in following actions of the game.

The server engine that you should implement must be able to process all the events, and for every summoned creature it must determine the number of the turn when this creature dies, or find out that it is still alive at the end of the game.

Also, the engine should correctly process the cases when the player tries to interact with the dead creatures: if the spell *Blessed Champion*, *Divine Spirit* or *Attack!* is applied to the dead creature, nothing should happen. If the spell *Molten Reflection* is applied to the dead creature, a new creature with the same parameters is summoned, but it is assumed to be killed at the turn of its creation.

Input

The first line of input contains an integer n — the number of turns in the game ($1 \le n \le 250\,000$).

The next n lines contain turn descriptions in the following format:

- 1 — apply a spell *Summon Creature*;
- 2 i — apply a spell *Blessed Champion* to the creature with id i;
- 3 i — apply a spell *Divine Spirit* to the creature with id i;
- 4 i — apply a spell *Molten Reflection* to the creature with id i;
- 5 i j — apply a spell *Attack!* to the creatures with ids i and j.

It is guaranteed that any creatures mentioned in the queries have already been summoned at the time of the query, but they can be already dead.

Output

The first line of output must contain one integer k — the number of creatures summoned in the game.

The next line must contain k integers $t_1, t_2, \ldots, t_k$. If the creature with id i is alive at the end of the game, t_i should be equal to -1, otherwise t_i should be equal to the number of the turn, when this creature was killed.

Example

standard input	standard output
16 1 2 1 3 1 1 5 1 2 3 1 1 3 3 3 3 4 1 5 1 3 3 3 5 1 3 5 4 3 5 4 3 4 1	5 13 5 14 -1 16

Note

The table below shows how the parameters of the creatures changed in the sample test.

turn	1	2	3	4	5
0	-	-	-	-	-
1	(1, 1)	-	-	-	-
2	(2, 1)	-	-	-	-
3	(2, 2)	-	-	-	-
4	(2, 2)	(1, 1)	-	-	-
5	(2, 1)	dead	-	-	-
6	(2, 2)	dead	-	-	-
7	(2, 2)	dead	(1, 1)	-	-
8	(2, 2)	dead	(1, 2)	-	-
9	(2, 2)	dead	(1, 4)	-	-
10	(2, 2)	dead	(1, 4)	(2, 2)	-
11	(2, 1)	dead	(1, 2)	(2, 2)	-
12	(2, 1)	dead	(1, 4)	(2, 2)	-
13	dead	dead	(1, 2)	(2, 2)	-
14	dead	dead	dead	(2, 1)	-
15	dead	dead	dead	(2, 1)	-
16	dead	dead	dead	(2, 1)	dead

Problem H. Linearization

Time limit:	2 seconds
Memory limit:	512 megabytes

Bitwise "and" of two non-negative integers is calculated as follows: write both numbers in binary, then the i-th binary digit of the result is equal to 1 if both arguments have the i-th digit equal to 1. For example, $(14 \text{ and } 7) = (1110_2 \text{ and } 0111_2) = 110_2 = 6$.

"Exclusive or" (xor) of two binary digits equals 1 if they are unequal, and 0 if they are equal. Thus, 0 xor 0 = 0, 0 xor 1 = 1, 1 xor 0 = 1 and 1 xor 1 = 0.

Parity function $P(x)$ for a non-negative integer x equals 1 if the binary notation of x has odd number of ones, and 0 if the binary notation of x has even number of ones. For example, $P(5) = P(101_2) = 0$, $P(7) = P(111_2) = 1$.

Consider a binary string whose length is a power of two: $s = s_0 s_1 \dots s_{n-1}$, where $n = 2^k$. We will call this string *linear*, if there is an integer x, $0 \le x < n$, and a binary digit b, such that for all i from 0 to $n - 1$ holds $s_i = P(i \text{ and } x) \text{ xor } b$.

For example, a string "`1100`" is linear: take $x = 2 = 10_2$ and $b = 1$.

- $s_0 = P(0 \text{ and } 2) \text{ xor } 1 = P(0) \text{ xor } 1 = 0 \text{ xor } 1 = 1$
- $s_1 = P(1 \text{ and } 2) \text{ xor } 1 = P(0) \text{ xor } 1 = 0 \text{ xor } 1 = 1$
- $s_2 = P(2 \text{ and } 2) \text{ xor } 1 = P(2) \text{ xor } 1 = 1 \text{ xor } 1 = 0$
- $s_3 = P(3 \text{ and } 2) \text{ xor } 1 = P(2) \text{ xor } 1 = 1 \text{ xor } 1 = 0$

Meanwhile, "`0001`" is not linear: whatever x we chose, we would have $P(0 \text{ and } x) = P(0) = 0$, therefore $b = 0$. We have $0 = P(1 \text{ and } x)$ and $0 = P(2 \text{ and } x)$, therefore $x = 0$. But $P(3 \text{ and } 0) = 0 \ne s_3 = 1$.

Consider a binary string. In one action you can take a continuous segment of digits and invert them: change all zeros to ones and vice versa. Call *hardness of linearization* of this string the minimal number of actions one needs to make it linear.

For example, the hardness of linearization for the string "`0001`" is 1: you can invert the left three digits to get the string "`1111`" which is linear with $x = 0$, $b = 1$. There are other ways to linearize it in one action.

You are given a string t and q queries (l_i, r_i). For each query, consider a substring of t from l_i-th digit to r_i-th digit, inclusive. Digits of t are numbered from left to right, starting with 0. It is guaranteed that the length of each query is a power of two. Calculate the hardness of linearization for every given substring.

Input

The first line of input contains a single integer m — the length of the string t ($1 \le m \le 200\,000$). The second line contains a binary string t of length m.

The next line contains integer q — the number of queries ($1 \le q \le 200\,000$). Each of the next q lines contains two integers, l_i and r_i ($0 \le l_i \le r_i < m$, $r_i - l_i + 1 \ge 2$, substring length is a power of two).

Output

For each query, print one integer: the hardness of linearization of the corresponding substring of t.

Example

standard input	standard output
8 00000101 3 0 7 2 5 0 3	2 1 0

Note

In the first query we need to linearize the whole string. This can be done, for example, by inverting the segment from 4-th to 6-th digit, getting the string "00001011", and then inverting the 5-th digit, getting "00001111" which is linear with $x = 4$ and $b = 0$.

In the second query, the string "0001'' can be linearized in one action, as described in the problem statement.

In the third query the string "0000" is already linear with $x = 0$, $b = 0$.

Problem I. Minimal Product

Time limit: 2 seconds
Memory limit: 512 megabytes

You are given an array of integers $a_1, \ldots, a_n$. Find two indices i and j such that $i < j$, $a_i < a_j$, and the product $a_i \cdot a_j$ is as small as possible.

Input

The input consists of several tests. The first line contains a single integer t — the number of tests ($1 \le t \le 10^4$). Each of the following t lines describes one test.

Each test is generated using the following algorithm. The test is described by integers n, l, r, x, y, z, b_1, b_2 ($2 \le n \le 10^7$, $-2 \cdot 10^9 \le l \le r \le 2 \cdot 10^9$, $0 \le x, y, z, b_1, b_2 < 2^{32}$), where n is the length of the array.

First, the sequence b_i of length n is generated. Elements b_1 and b_2 are given. For $i > 2$ let $b_i = (b_{i-2}x + b_{i-1}y + z) \bmod 2^{32}$. For each i between 1 and n, $a_i = (b_i \bmod (r - l + 1)) + l$ (thus, $-2 \cdot 10^9 \le a_i \le 2 \cdot 10^9$).

It is recommended to use 64-bit integers to generate the sequence to avoid integer overflow.

The sum of n in all tests does not exceed $2 \cdot 10^7$.

Output

For each test, print the smallest possible product $a_i \cdot a_j$ in a separate line. If there are no such i and j that $i < j$ and $a_i < a_j$, print "`IMPOSSIBLE`".

Example

standard input	standard output
2 4 -5 5 11 13 17 0 3 5 0 100 0 1 0 42 42	-15 IMPOSSIBLE

Note

Let us consider the generation of the array in the first test.

First, the sequence b is generated.

- $b_1 = 0$
- $b_2 = 3$
- $b_3 = (11 \cdot 0 + 13 \cdot 3 + 17) \bmod 2^{32} = 56$
- $b_4 = (11 \cdot 3 + 13 \cdot 56 + 17) \bmod 2^{32} = 778$

Then it is used to generate a.

- $a_1 = (0 \bmod (5 - (-5) + 1)) + (-5) = (0 \bmod 11) - 5 = -5$
- $a_2 = (3 \bmod 11) - 5 = -2$
- $a_3 = (56 \bmod 11) - 5 = -4$
- $a_4 = (778 \bmod 11) - 5 = 3$

Thus, $a = [-5, -2, -4, 3]$. The answer is $-5 \cdot 3 = -15$.

In the second test the array is $[42, 42, 42, 42, 42]$.

Problem J. Two Prefixes

Time limit: 1 second
Memory limit: 512 megabytes

Misha didn't do his math homework for today's lesson once again. As a punishment, his teacher Dr. Andrew decided to give him a hard, but very useless task.

Dr. Andrew has written two strings s and t of lowercase English letters at the blackboard. He reminded Misha that *prefix* of a string is a string formed by removing several (possibly none) of its last characters, and a *concatenation* of two strings is a string formed by appending the second string to the right of the first string.

The teacher asked Misha to write down on the blackboard all strings that are the concatenations of some non-empty prefix of s and some non-empty prefix of t. When Misha did it, Dr. Andrew asked him how many distinct strings are there. Misha spent almost the entire lesson doing that and completed the task.

Now he asks you to write a program that would do this task automatically.

Input

The first line contains the string s consisting of lowercase English letters. The second line contains the string t consisting of lowercase English letters.

The lengths of both string do not exceed 10^5.

Output

Output a single integer — the number of distinct strings that are concatenations of some non-empty prefix of s with some non-empty prefix of t.

Examples

standard input	standard output
aba aa	5
aaaaa aaaa	8

Note

In the first example, the string s has three non-empty prefixes: {a, ab, aba}. The string t has two non-empty prefixes: {a, aa}. In total, Misha has written five distinct strings: {aa, aaa, aba, abaa, abaaa}. The string abaa has been written twice.

In the second example, Misha has written eight distinct strings: {aa, aaa, aaaa, aaaaa, aaaaaa, aaaaaaa, aaaaaaaa, aaaaaaaaa}.

Problem K. Right Expansion Of The Mind

Time limit:	2 seconds
Memory limit:	512 megabytes

One day n people decided to expand their mind.

Initially, the mind of the i-th person is two strings s_i and t_i consisting of lowercase English letters. After the expansion, the mind of the person becomes an infinite to the right string $w_i = s_i + t_i + t_i + \cdots$. That is, w_i is a concatenation of s_i and the infinite number of t_i. For example, if $s_i =$ `mi` and $t_i =$ `nd`, the corresponding $w_i =$ `mindndndnd`$\ldots$.

Two persons with expanded mind are *interested* in each other, if the mind of the first person forms a subsequence of the mind of the second person, and the mind of the second person forms a subsequence of the mind of the first person. The infinite string a is called a subsequence of the infinite string b if there is an infinite sequence of indices $1 \le i_1 < i_2 < i_3 < \ldots$, such that for each j it is $a_j = b_{i_j}$. For example, the infinite string "`baaa`$\ldots$" is a subsequence of an infinite string "`cababababa`$\ldots$".

After having the mind expanded, people decided to split into groups in such way, that any two persons in one group are interested in each other. Your task is to split them into the groups, so that the number of groups is minimal possible.

Input

The first line contains one integer n — the number of people that have decided to get their mind expanded ($1 \le n \le 100\,000$).

Each of the following n lines contains two non-empty strings s_i and t_i, consisting of lowercase English letters. They represent the minds of every person before the mind expansion.

It is guaranteed that the total length of all strings doesn't exceed $1\,000\,000$.

Output

The first line must contain one integer — the minimum number of groups.

Then print each group in the following way: first print the number of people in this group and then the indices of them.

Each of the indices must be printed exactly once. The groups and indices of the people in the group can be printed in any order. If there are multiple ways to split the people, you can print any of them.

Examples

standard input	standard output
5 ab ab ababab ab a abb x y z w	3 3 1 2 3 1 4 1 5
3 kokoko tlin koko kotlin ko kokotlin	2 1 1 2 2 3

Note

In the first example the expanded minds of the people look as follows:

- "`abababab...`"
- "`abababab...`"
- "`aabbabbabb...`"
- "`xyyyyyy...`"
- "`zwwwwww...`"

Neither person 4 nor 5 are interested in anybody else. However, the first three people show pairwise interest in each other. Hence it is possible to have the groups $\{1, 2, 3\}$, $\{4\}$, $\{5\}$.

Problem L. Berland University

Time limit:	1 second
Memory limit:	512 megabytes

There are t students studying in the best university of Berland. They only study programming in Berland, so there is only one subject. Each student must attend the lectures.

The entire course consists of n lectures. It is known that a student who visits at least k of them will pass the course.

There are only two auditoriums in the university, one has space for a people and the other one — for b people. To make it comfortable, the administration decided that in odd weeks the lectures will be in the first auditorium, and in the even weeks — in the second auditorium. So the first lecture will be in the 1-st auditorium, the second lecture in the 2-nd one, the third in the 1-st one again, and so on.

The sizes of the auditoriums are small so it might not be possible for all students to attend at least k lectures. They ask you to count the maximum number of students that can pass the course.

Input

The first line contains five integers:

- t — the number of students;
- n — the number of lectures;
- a — the size of the first auditorium;
- b — the size of the second auditorium;
- k — the minimal number of lectures to pass the course.

The limits are: $1 \le t, n, a, b, k \le 10^9$.

Output

Print a single integer — the maximal number of students that can attend at least k lectures and thus pass the course.

Examples

standard input	standard output
10 3 4 4 3	4
10 3 4 4 5	0
100000 100000 100000 100000 1	100000
5 4 5 3 3	5
100 9 6 3 6	7

Note

In the fourth sample, 5 students can pass the course. Here's one possible strategy:

1. Students 1, 2, 3, 4, 5 visit the first lecture.
2. Students 1, 3 visit the second lecture.
3. Students 1, 2, 3, 4, 5 visit the third lecture.
4. Students 2, 4, 5 visit the fourth lecture.

This way each of these 5 students can attend at least 3 lectures.

Problem M. The Pleasant Walk

Time limit: 1 second
Memory limit: 512 megabytes

There are n houses along the road where Anya lives, each one is painted in one of k possible colors.

Anya likes walking along this road, but she doesn't like when two adjacent houses at the road have the same color. She wants to select a long segment of the road such that no two adjacent houses have the same color.

Help Anya find the longest segment with this property.

Input

The first line contains two integers n and k — the number of houses and the number of colors ($1 \le n \le 100\,000$, $1 \le k \le 100\,000$).

The next line contains n integers $a_1, a_2, \ldots, a_n$ — the colors of the houses along the road ($1 \le a_i \le k$).

Output

Output a single integer — the maximum number of houses on the road segment having no two adjacent houses of the same color.

Example

standard input	standard output
8 3 1 2 3 3 2 1 2 2	4

Note

In the example, the longest segment without neighboring houses of the same color is from the house 4 to the house 7. The colors of the houses are $[3, 2, 1, 2]$ and its length is 4 houses.