Problem A. At random

Source file name:	`a.c, a.cpp, a.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Fidel I. Schaposnik - Universidad Nacional de La Plata`

There are many and very different card games, their origin going back to ancestral times. Sometimes it can be surprising that they continue to provide entertainment after so many centuries of being played with the same rules, but then we should remember that each game is essentially different from all the other ones played in the history of humankind, given the huge amount of possible ways to order the cards before the beginning of each game. Indeed, few games are any fun if we always use the cards in the same order, or if there is a correlation between successive cards that allows us to guess the order in which they will appear. This is the reason why it is customary to shuffle the cards before starting each game, and for this same reason we ask you to make a program to check that a sequence of cards has been properly shuffled.

To simplify the problem, we will concentrate only on decks of Spanish cards, which consist of 48 different playing cards. Each card is identified by a value, which is a number from 1 to 12, and a suit, which can be "clubs", "cups", "golds" or "swords". However, because we don't want to excessively simplify your task, we will take into account that not all games use all 48 cards of the deck. Given a sequence of N cards, we say it has been properly shuffled if there are no two successive cards sharing the same value or suit. Otherwise, we say the deck has not been properly shuffled. Can you help us decide if a given sequence has been properly shuffled?

Input

The first line of each test case contains an integer N, representing the number of cards that are used in the game we are considering ($2 \leq N \leq 48$). Each of the following N lines contains the description of a card in the sequence we want to analyze, given by an integer V representing its value ($1 \leq V \leq 12$) and a character P representing its suit: "b" for clubs, "c" for cups, "o" for golds and "e" for swords. All the cards in the input are different, and they are given in the input in the same order they appear in the sequence.

Output

Print a single line containing a character indicating if the sequence of cards given in the input has been properly shuffled or not. The character should be a "B" if it has been properly shuffled, and an "M" otherwise.

Example

Input	Output
4 1 b 2 c 3 e 4 o	B
3 1 b 2 b 3 c	M
3 1 b 1 c 2 e	M
32 5 c 2 b 4 e 3 o 12 b 1 c 7 e 6 c 12 e 4 o 1 b 6 o 3 e 12 o 11 e 12 c 5 o 10 b 9 o 3 c 4 b 11 c 8 e 9 c 1 e 4 c 8 b 2 o 6 b 9 e 7 b 5 e	B

Problem B. Balanced base-3

Source file name:	`b.c, b.cpp, b.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Fidel I. Schaposnik - Universidad Nacional de La Plata`

Throughout history many different numbering systems have been developed. Some, like Roman numerals, have now been almost completely abandoned due to their lack of convenience. Other even more exotic numbering systems are only ever used in certain applications, such as the factoradic numbers in the context of permutation numbering. In this problem we will consider a numbering system known as balanced base-3, which naturally appears in the analysis of diverse mathematical problems related to balance scales.

The balanced base-3 system is similar to the decimal or base-10 system we are familiar with in that it is positional. Positional numbering systems have digits whose relative order determines which power of the base accompanies them. For example, in base-10 we have

$$123 = \mathbf{1} \times 10^2 + \mathbf{2} \times 10^1 + \mathbf{3} \times 10^0.$$

In standard positional systems, the allowed digits are the numbers from 0 to $B-1$, where B is the base of the corresponding system. Therefore, the number 123 in base-10 is written in standard base-3 as "11120" because

$$\mathbf{1} \times 3^4 + \mathbf{1} \times 3^3 + \mathbf{1} \times 3^2 + \mathbf{2} \times 3^1 + \mathbf{0} \times 3^0 = 123.$$

Balanced base-3 is different from standard base-3 only in the fact that the allowed digits are 0, 1 and -1, which we will respectively denote as "0", "+" and "-". Then, the number 123 in base-10 is written as "`+----0`" in balanced base-3, for we have

$$\mathbf{1} \times 3^5 + (-\mathbf{1}) \times 3^4 + (-\mathbf{1}) \times 3^3 + (-\mathbf{1}) \times 3^2 + (-\mathbf{1}) \times 3^1 + \mathbf{0} \times 3^0 = 123.$$

Conversion from numbers in base-10 to balanced base-3 is a mechanical and somewhat tedious process, so we require a program to do it for us. Can you help?

Input

Each test case is described by a single line containing a positive integer number N, the number in base-10 that we want to write in balanced base-3 ($1 \leq N \leq 1000$).

Output

Print a single line containing a string composed solely of the characters "0", "+" and "-" and not beginning with "0", representing the digits of the number N in balanced base-3. Note that the restriction that the string does not start with a "0" ensures that this representation is unique.

Example

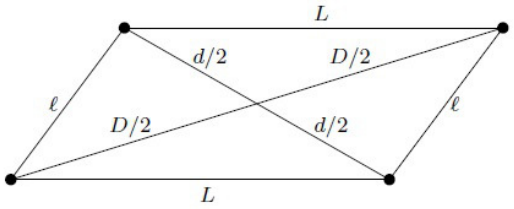
Input	Output
123	`+----0`
729	`+000000`

Problem C. Constellation of the parallelogram

Source file name:	`c.c, c.cpp, c.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Pablo A. Heiber - Universidad de Buenos Aires`

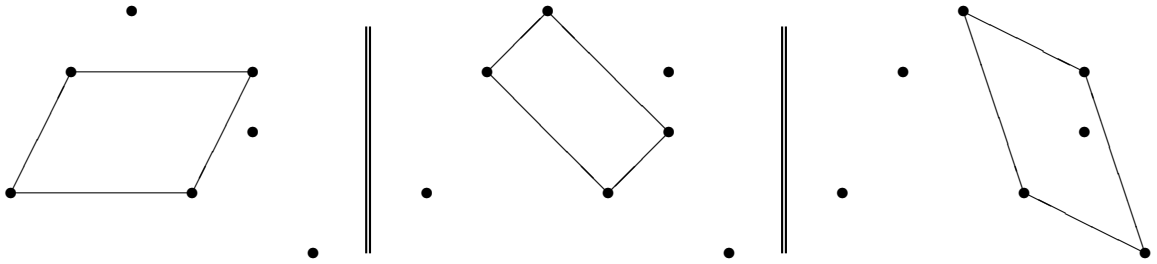
One morning, when Cyrus Samsa woke from troubled dreams, he found himself lying on his bed, but not transformed into a monstrous insect. However, when he looked up at the still starry sky, he found that the visual setting of the cosmos showed four stars positioned as the vertices of a perfect parallelogram. Being a big geometry enthusiast, he called those four stars "constellation of the parallelogram".

A parallelogram is a four-sided polygon with its pairs of opposite sides being of identical length. Equivalently, a parallelogram is a convex quadrilateral whose diagonals intersect at their midpoints. The following figure illustrates both definitions, for a parallelogram with sides of lengths L and l, and diagonals of lengths D and d.



Unfortunately, astronomers still do not agree on which were the stars that Cyrus was looking at. The problem is that on each image obtained of the sky's configuration, there are usually many sets comprising four stars that are the vertices of a parallelogram.

A starry sky is represented on an image as a set of points in the Cartesian plane, each of them corresponding to a star. The following figure shows three copies of the same image, each illustrating one of the three sets of four stars which are the vertices of a parallelogram, which are marked in the figure with solid lines.



Your task is to safeguard the good name of astronomers by calculating, given an image, the number of parallelograms contained in it.

Input

The first line contains an integer N indicating the number of stars on the image to be analyzed ($4 \leq N \leq 1000$). The following N lines describe each of the stars on the image with two integers X_i and Y_i, indicating respectively the X and Y coordinates of the star on the Cartesian plane ($-10^8 \leq X_i$, $Y_i \leq 10^8$ for $i = 1, 2, \ldots, N$). You may assume that no two stars on the image are on the same position.

Output

Print a single line containing a single integer representing the number of sets of four points on the input image that are the vertices of a parallelogram.

Example

Input	Output
7 0 1 1 3 2 4 3 1 4 2 4 3 5 0	3
11 0 0 0 1 0 -1 1 0 1 1 1 -1 -1 0 -1 1 -1 -1 0 2 1 2	44
4 0 0 0 1 0 2 0 3	0
6 -100000000 100000000 100000000 -100000000 -100000000 -100000000 100000000 100000000 1 1 -1 -1	2

Problem D. Fractal domino

Source file name:	`d.c, d.cpp, d.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Fidel I. Schaposnik - Universidad Nacional de La Plata`

A question we may ask ourselves if we have a lot of free time (or if we are mathematicians, which is practically the same thing) is the following: given a board of $N \times N$ square cells, in how many ways can we cover it with dominoes so that the pieces do not overlap and there are no cells left uncovered? Dominoes are rectangles composed of two adjacent square cells, and can be either horizontal (1×2) or vertical (2×1). For example, there are only two ways to cover a 2×2 board, but there are 12.988.816 ways to cover an 8×8 board.

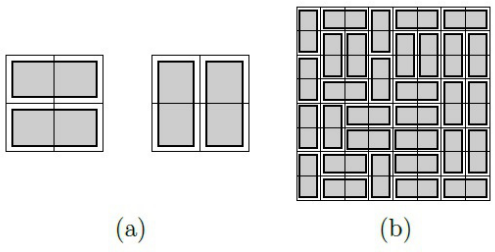


Figure 1: (a) The two ways to cover a 2×2 board with horizontal dominoes (left) and vertical dominoes (right); (b) a possible covering of an 8×8.

For those with even more free time, there is a variant of this question in which we consider a board with some of its cells occupied, so that we should count the number of coverings in which the domino pieces cover only the unoccupied cells without overlapping or leaving holes. For example, there are two ways to cover a 3×3 board in which the central cell is occupied, and 75.272 ways to do so in a 7×7 board.

In this problem we will turn our attention to a variant of this variant of the original question, specifically thought for those of you with too much free time. We will again consider an $N \times N$ board in which some cells can be occupied, but we now want to cover it using fractal dominoes of order K.

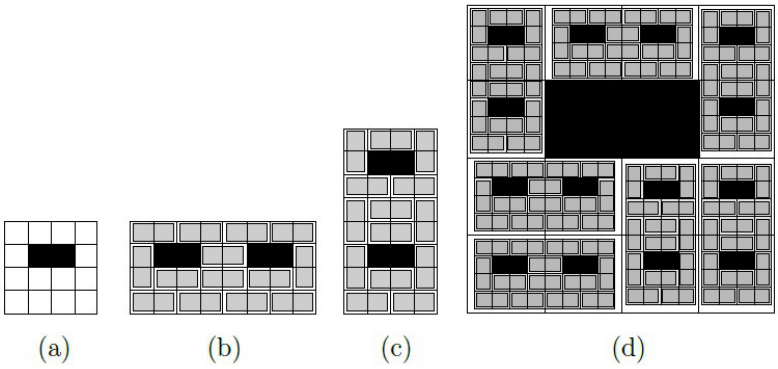


Figure 2: (a) A 4×4 board with two occupied cells; (b) one of the 89 horizontal dominoes of order one for this board; (c) one of the 52 vertical dominoes of order one for the same board; (d) a covering of the board in (a) with the fractal dominoes of order one in (b) and (c).

Given an $N \times N$ board, a fractal domino of order $K = 0$ is a regular domino piece, i.e. a rectangle of 1×2 or 2×1 cells. A fractal domino piece of order $K > 0$ consists of a regular domino in which each

cell is a copy of the given $N \times N$ board, and the piece as a whole is covered by fractal dominoes of order $K-1$.

Note that in general for $K > 1$ there can be more than one fractal domino piece of order $K-1$ of each type (horizontal or vertical). In this case, we assume that when using a domino piece it is chosen with uniform probability among all possible fractal domino pieces of the same type and order. In a similar fashion, if there is more than one possible covering for a given board or fractal domino, we will assume all of them are equally probable.

When covering a board with fractal dominoes of order K one will use a certain amount of pieces of order K, a larger amount of pieces of order $K-1$, an even larger amount of pieces of order $K-2$, and so on. This will go on up to the point where one uses a possibly huge amount of pieces of order zero, i.e. regular domino pieces with nothing inside. For example, in Figure 2d the board is covered with seven fractal dominoes of order one and 98 fractal dominoes of order zero.

The task in this problem is to calculate the expected proportion of fractal dominoes of order zero (i.e. regular pieces) that are vertical, if we assume that all coverings of a given board with fractal dominoes of a given order K are equally probable.

Input

The first line of each test case contains two integers N and K, representing the size of the board and the order of the fractal dominoes that we want to use to cover it, respectively ($2 \leq N \leq 8$ and $0 \leq K \leq 10^9$). The following N lines contain N numbers each, being the j-th number in the i-th line a 1 if the cell in row i and column j of the board is occupied, and 0 otherwise. The given board is not completely occupied, and it is always possible to cover it using dominoes.

Output

For each test case, print a single line containing a rational number representing the expected proportion of fractal dominoes of order zero that are vertical, when we consider that all coverings of the given board with fractal dominoes of order K are equally probable. Print the result using exactly 5 decimal digits after the decimal point, rounding if necessary.

Example

Input	Output
4 2 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0	0.50750

Problem E. Erdös et al

Source file name:	`e.c, e.cpp, e.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Pablo A. Heiber - Universidad de Buenos Aires`

Paul Erdös was a Hungarian mathematician of the 20th century who reached the highest levels of recognition. So much so that it is considered an honour not only having written an article with him, but also having shared authorship of a publication with one of his co-authors. Moreover, writing an article with someone who wrote an article with someone who wrote an article with Erdös is an important aspiration of many young researchers.

A natural consequence of such honours assignment, at least within the context of formal sciences, is the emergence of Erdös numbers. Erdös is the only person with an Erdös number of 0, and for any other person p, his/her Erdös number n is defined by the shortest sequence of articles $a_1, \ldots, a_n$ such that Erdös is one of the authors of article a_1, p is one the authors of article an, and every pair of consecutive articles a_i and a_{i+1} (for $i = 1, 2, \ldots, N-1$) have at least one author in common. If no sequence of articles linking Erdös and p exists, we shall say that p's Erdös number is undefined.

Your task in this problem is to compute Erdös numbers based only on a corpus of articles and authors given as input. Unfortunately, current science demands scientists to increase very rapidly the number of articles they write, causing both the total number of articles and the number of authors per article to be huge. Of course, this reality is an obstacle that a correct solution to this problem should be able to handle.

Input

The first line contains two integers A and N, where A represents the number of articles in the corpus to be analysed and N the number of people who appear in it (where $2 \leq A, N \leq 10^5$). People are identified with integers between 1 and N, and Erdös will always be the person identified with number 1.

Each of the following A lines describes an article. Each of these lines begins with an integer C representing the number of authors of the article ($2 \leq C \leq 10^5$), and then there are C distinct integers $P_1, P_2, \ldots, P_C$ representing the identifiers of the authors of the article ($1 \leq P_i \leq N$ for $i = 1, 2, \ldots, C$). The sum of the number of authors over all articles does not exceed 10^5.

Output

For each test case you must print three integers D, M and S, where D represents the number of people on the corpus who have a well-defined Erdös number, M is the maximum Erdös number of one of those people and S is the sum of all the Erdös numbers belonging to people who have a well-defined Erdös number.

Example

Input	Output
3 5 2 1 5 3 5 2 3 3 3 2 4	5 3 8
5 11 2 10 11 4 1 3 5 7 6 2 3 4 5 6 7 4 3 5 7 9 3 8 1 5	9 2 12
6 31 9 1 2 3 15 20 25 30 9 10 10 2 25 7 9 3 11 12 13 14 4 10 11 12 13 14 4 16 17 18 19 5 2 5 7 9 21 22 23 24 26 27 28 29 7 3 22 6 21	29 4 63

Problem F. String fertilization

Source file name:	`f.c, f.cpp, f.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Mario Ynocente Castro`

Strings are like plants in that they require a lot of loving care to grow. In this problem we will follow the evolution of a garden with N strings during a period covering T seasons. The strings in the garden are numbered from 1 to N, and are all initially empty. Each season we will perform two tasks in our garden:

- At the beginning of the season, we may prune the garden by deleting the C characters on the rightmost end of each of the N strings in the garden.
- After the pruning is done, we fertilize the garden so that each of the N strings grows by appending one character (possibly different for each string) to its rightmost end.

At the end of the season, a good string gardener always takes a moment to contemplate his/her work. In order to do this, we take a number P from 1 to N and then dedicate ourselves to appreciate the beauty of the string that stands at position P when we sort the N strings in the garden alphabetically from smallest to largest (breaking draws arbitrarily by putting first the strings identified with smaller numbers).

These moments of contemplation should be a well-deserved resting time for the gardener, so we don't want to waste time sorting the strings in the garden to identify the one we want to appreciate. Can you help us find it?

Input

The first line contains two integer numbers N and T, representing the number of strings in the garden and the number of seasons we follow their evolution, respectively ($2 \leq N \leq 100$ and $1 \leq T \leq 10^4$). The following T lines describe one season each, in the same order in which they take place.

The description of each season consists of a number C, a string S and another number P ($1 \leq P \leq N$). The number C is non-negative and represents the number of characters that are deleted during the pruning period at the beginning of the corresponding season (so it can be zero in case that no pruning is performed in that season). The string S contains exactly N characters $s_1, s_2, \ldots, s_N$, being the i-th character s_i the one that should be appended to the rightmost end of the string identified by the number i (s_i is a lower-case letter of the English alphabet for $i = 1, 2, \ldots, N$). Finally, the number P represents the position of the string we would like to appreciate at the end of the season, when we sort the N strings in the garden as explained in the problem statement.

Output

Print T lines, one for each season described in the input. The i-th line should contain the number identifying the string we want to appreciate at the end of the i-th season, for $i = 1, 2, \ldots, T$.

Example

Input	Output
2 4 0 aa 1 0 ba 1 1 ba 2 2 aa 2	1 2 1 2
26 26 0 abcdefghijklmnopqrstuvwxyz 1 1 bcdefghijklmnopqrstuvwxyza 2 1 cdefghijklmnopqrstuvwxyzab 3 1 defghijklmnopqrstuvwxyzabc 4 1 efghijklmnopqrstuvwxyzabcd 5 1 fghijklmnopqrstuvwxyzabcde 6 1 ghijklmnopqrstuvwxyzabcdef 7 1 hijklmnopqrstuvwxyzabcdefg 8 1 ijklmnopqrstuvwxyzabcdefgh 9 1 jklmnopqrstuvwxyzabcdefghi 10 1 klmnopqrstuvwxyzabcdefghij 11 1 lmnopqrstuvwxyzabcdefghijk 12 1 mnopqrstuvwxyzabcdefghijkl 13 1 nopqrstuvwxyzabcdefghijklm 14 1 opqrstuvwxyzabcdefghijklmn 15 1 pqrstuvwxyzabcdefghijklmno 16 1 qrstuvwxyzabcdefghijklmnop 17 1 rstuvwxyzabcdefghijklmnopq 18 1 stuvwxyzabcdefghijklmnopqr 19 1 tuvwxyzabcdefghijklmnopqrs 20 1 uvwxyzabcdefghijklmnopqrst 21 1 vwxyzabcdefghijklmnopqrstu 22 1 wxyzabcdefghijklmnopqrstuv 23 1 xyzabcdefghijklmnopqrstuvw 24 1 yzabcdefghijklmnopqrstuvwx 25 1 zabcdefghijklmnopqrstuvwxy 26	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Problem G. Gallantry

Source file name:	`g.c, g.cpp, g.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Leopoldo Taravilse - Universidad de Buenos Aires`

Every weekend, Germán and Gianina play a football match with their friends. This Saturday, both German and Gianina are injured, so they will participate in the game as coaches.

To choose the teams they will use the following procedure: of the N friends they have in common, each of them is going to pick $N/2$ players (N is even), with the selection process taking $N/2$ turns. On every turn, each of them chooses a different player among the ones who have not been selected yet, and this player is assigned to his/her team. Germán won the initial coin flip so he gets to choose first on every turn. However, to be gallant Germán can let Gianina go first on some turns if he wants so.

For example, if $N = 6$ there are $N/2 = 3$ turns. Suppose Germán decides to yield his right to choose first on the second and third turns, but not on the first one. In that case, the order of selection of players would be:

$$\underbrace{\text{Germán} - \text{Gianina}}_{\text{turn 1}} - \underbrace{\text{Gianina} - \text{Germán}}_{\text{turn 2}} - \underbrace{\text{Gianina} - \text{Germán}}_{\text{turn 3}}.$$

As we all know, football is a sport ruled by logic. Every friend of Germán and Gianina has a score that indicates how well they play. If the sum of scores of Germán's team is strictly greater than the sum of scores of Gianina's team, then Germán's team will win the match. If the sum of scores is the same for both teams, the match ends in a draw. Otherwise, Gianina's team will be the winner.

Germán wants to be gallant, but he is even more interested in winning the game. Therefore, he wants to yield his turn as many times as possible so that he can still win the game anyway. Gianina also wants to win the game, therefore each time they choose a player they will both do it in a way that maximizes their chances of winning.

Input

The first line contains an integer N representing the number of players to choose from (with $2 \leq N \leq 1000$ and N even). The second line contains N integers $P_1, P_2, \ldots, P_N$ representing the scores of each player ($1 \leq P_i \leq 1000$ for $i = 1, 2, \ldots, N$).

Output

Print a line containing a single integer that represents how many times Germán can yield his turn in such a way that his team's victory is guaranteed. If Germán can not ensure a victory, you must print the number -1.

Example

Input	Output
6 7 8 2 10 1 4	1
6 7 8 2 10 1 3	2
4 60 95 100 65	0
10 10 10 10 10 10 10 10 10 10 10	-1

Problem H. Rush hour

Source file name:	`h.c, h.cpp, h.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Nicolás Álvarez - Universidad Nacional del Sur`

Nlogonia is a very organized city, where the houses of the inhabitants are all located on the East End of the city and their workplaces are located on the West End.

Each working day, people go to their workplace in the West and at the end of the day they return to their homes in the East. They rely on the urban rail system of Nlogonia for transportation.

The train company offers N different services. Each of them makes a journey that links a station on the West to a station on the East. There are exactly N stations on the West and N stations on the East. At both ends, the stations are numbered from 1 to N following North-South order. The station furthest to the North is identified with the number 1 and the station furthest to the South with the number N. On both ends, each station belongs to exactly one service.

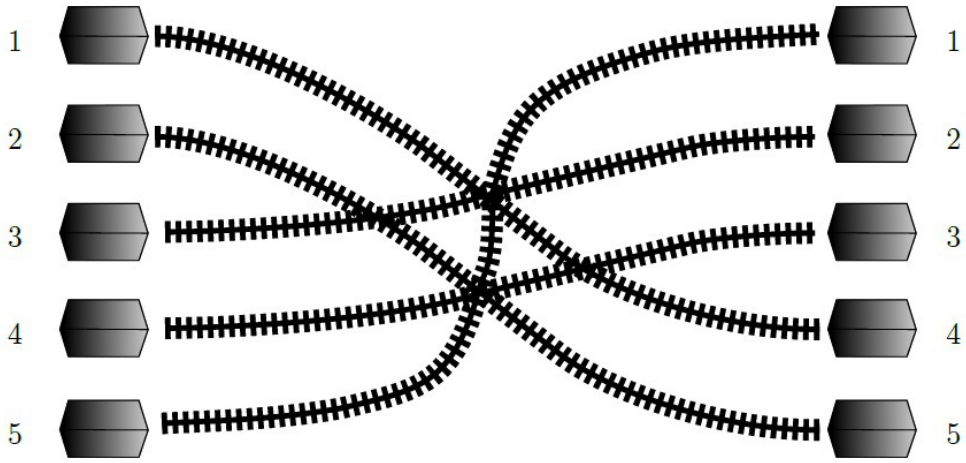


Figure 1: Situation with $N = 5$ stations corresponding to the first sample input.

When the workday ends, the Nlogonians leave their workplaces eager to return home. This results in heavy traffic over the rail system usually known as "rush hour traffic".

As shown in Figure 1 above, some train services cross over their paths. Two paths cross each other if and only if the relative order of the trains in the North-South direction is different when leaving the West station than when they reach the East station.

To avoid accidents, Nlogonia's train company decided to schedule the trains' departures in shifts so that there are no two trains whose routes cross leaving on the same shift. They want to avoid delays, so their goal is to come up with a schedule such that the number of shifts is minimized. Luckily, there is exactly one train of each service leaving on rush hour.

Continuing with the example of the previously shown Figure 1, the service departing from station 5 intersects with all the other services, therefore a shift must be scheduled exclusively for its departure. The remaining four services cannot be scheduled on a single shift as there are some crossings among them. However, it is possible to group them into two shifts, one for trains leaving stations 1 and 2, and another one for trains leaving stations 3 and 4. Thus, a total of 3 shifts will be necessary to avoid all risk of collisions.

You will be responsible for this task: given the descriptions of the N train services in Nlogonia, you must determine the minimum number of turns in which you can plan train departures avoiding all possible risk of accidents.

Input

The first line of each test case contains an integer N indicating the number of train services in Nlogonia ($1 \leq N \leq 10^5$). The second line contains N different integers $E_1, E_2, \ldots, E_N$ ($1 \leq E_i \leq N$), indicating that the train leaving the i-th station on the West should get to station E_i on the East, for $i = 1, 2, \ldots, N$.

Output

Print a line containing a single integer representing the minimum number of turns in which the services can be planned.

Example

Input	Output
5 4 5 2 3 1	3
3 1 2 3	1
9 9 4 2 7 8 3 5 6 1	4

Problem I. Stapled intervals

Source file name:	`i.c, i.cpp, i.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Fidel I. Schaposnik - Universidad Nacional de La Plata`

Two natural numbers n and m are said to be coprime if their greatest common divisor is the number 1. In other words, n and m are coprime if there is no integer $d > 1$ such that d exactly divides both n and m. A finite set of two or more consecutive natural numbers is called a "stapled interval" if there is no number in it that is coprime to all other numbers in the set.

Given a range $[A, B]$, we would like to count the number of stapled intervals completely contained in it. I.e., we want to know how many different pairs (a, b) exist such that $A \leq a < b \leq B$ and the set $\{a, a+1, \ldots, b\}$ is a stapled interval.

Input

The first line contains an integer P representing the number of questions you should answer ($1 \leq P \leq 1000$). Each of the following P lines describes a question, and contains two integer numbers A and B representing the borders of the range $[A, B]$ in which we want to count stapled intervals ($1 \leq A \leq B \leq 10^7$).

Output

Print P lines, each with a single integer number. For $i = 1, 2, \ldots, P$ the number in the i-th line represents the number of stapled intervals completely contained in the range $[A, B]$ corresponding to the i-th question.

Example

Input	Output
4 2184 2200 2185 2200 2184 2199 1 100000	1 0 0 13

Problem J. Distracted judges

Source file name:	`j.c, j.cpp, j.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Leopoldo Taravilse - Universidad de Buenos Aires`

This problem was included in this contest at the last minute, replacing another one that the jury could not fully prepare in time. We will not tell you much about the original problem because we will use it for the contest next year. We will only say that it was called "Playing with lists" because it deals with integer lists. The input for the problem was formed by L lists of integers, and its format was:

- a line with a positive integer N_1 indicating the amount of numbers in the first list;
- N_1 lines each representing an integer in the first list;
- a line with a positive integer N_2 indicating the amount of numbers in the second list;
- N_2 lines each representing an integer in the second list;
- ...
- a line with a positive integer N_L indicating the amount of numbers in the last list;
- N_L lines each representing an integer in the last list.

In an unfortunate accident, Joaquin -a member of the jury- erased the first line of each input file (the one which contains N_1). We need your help to restore the input files so we can use the problem next year. Given the file without the first line, we ask you to tell us which are the possible values of N_1 such that, if we add a line with that value at the beginning, the resulting file will be a valid input according to the description given above.

Input

The first line contains an integer T ($1 \leq T \leq 10^5$) which indicates the number of lines remaining in the input file. Each of the next T lines contains an integer R_i which represents the number left on the i-th line of the input file after the accident ($1 \leq R_i \leq 10^5$ for $i = 1, 2, \ldots, T$).

Output

Print a line for each possible value of N_1. Print all possible answers in increasing order.

Example

Input	Output
2 1 1	2
5 3 1 2 4 5	2 5
10 1 2 3 4 5 6 7 8 9 10	1 4 10

Problem K. Encryption kit

Source file name:	`k.c, k.cpp, k.java`
Input:	`Standard`
Output:	`Standard`
Setter(s):	`Walter Erquínigo - Facebook`

Karina is developing a powerful encryption protocol. To make it available to a wider audience, she wants to implement a tool-kit to make it easier to use. In particular, she requires a tool capable of efficiently performing a complicated operation over strings.

The positions in a string are numbered from left to right with consecutive natural numbers from 1 to the length of the string. The operation Karina needs to implement is specified by four positions in the string $i \leq j < k \leq l$, and consists of three steps:

1. interchange the substring that goes from position i to position j inclusive, with the substring that goes from position k to position l, inclusive;
2. reverse both substrings individually;
3. change all the characters of each substring into the next character in the alphabet. This is, every "a" turns into a "b", every "b" turns into a "c", etc. For the purposes of this operation we consider the alphabet to be circular, so that every "z" turns into an "a" after the operation is performed on it.

For example, let's take the string "alazareselfacil" and the positions $i = 3$, $j = 5$, $k = 8$ and $l = 15$, so that the two affected substrings are "aza" and "selfacil". After the first step (interchanging) the resulting string is "alselfacilreaza". After the second step (reversing) the result is "allicaflesreaza". Finally, after the third step (changing the characters to the ones following them) we obtain "almjdbgmftrebab".

The protocol developed by Karina applies the operation described above on a string, then applies another operation on the resulting string, and so on and so forth, always applying new operations on the last obtained result. Karina needs to know what is the resulting string after all operations are performed, but unfortunately her rudimentary knowledge of the programming language R is only enough to implement a very inefficient algorithm. You are therefore required to implement a version that can efficiently handle many operations over long strings.

Input

The first line of each test case contains the initial string S, composed of between 2 and 10^5 lower-case letters of the English alphabet, and an integer N, representing the number of operations that are to be performed ($1 \leq N \leq 10^5$).

The following N lines contain the description of one operation each, given by four integers I, J, K and L representing the four positions in the string for the operation explained above ($1 \leq I \leq J < K \leq L \leq |S|$ with $|S|$ the length of the string S).

Output

Print a single line containing a string representing the resulting string obtained after successively applying all the operations in the input to the given initial string.

Example

Input	Output
alazareselfacil 1 3 5 8 15	almjdbgmftrebab
alazareselfacil 2 3 5 8 15 3 5 8 15	alcbcfsugnbgekn
aa 1 1 1 2 2	bb
zabcdefghi 5 1 1 10 10 1 5 6 10 2 4 7 9 1 1 2 10 1 8 9 10	defghgjklm