Baby Bob's Coloring Book (circlecoloring)

Baby Bob celebrated his birthday recently, and just as you would expect, he received an extraordinary amount of presents from his family. His favorite gift is a coloring book, which contains colorless images that he can fill with colors however he wants to. Baby Bob's absolute favorite images do not depict childish things like dogs and birds but rather display more abstract objects like triangles, circles, and so on.

Figure 1: Some pages from Baby Bob's coloring book.

Today, Baby Bob has discovered a nice image. The image contains N empty circles arranged over the perimeter of a large circle. The circles are numbered from 0 to $N-1$ clockwise. There are line segments connecting circle i and circle $i+1$ for each $i = 0 \ldots N-2$, as well as circle $N-1$ and circle 0.

Baby Bob liked this picture and decided to modify it a bit. He drew M additional straight line segments. These new segments satisfy the following criteria:

- Each new line segment connects two circles that were not connected before.
- No two line segments intersect each other.
- All circles connected by the new segments are distinct, i.e., each circle is connected to **at most one** other circle by a new line.

Baby Bob wants to color the circles in his image. He only has crayons of two colors, red and blue. He wants to avoid coloring circles connected by line segments with the same color, but he is mature enough to accept that sometimes he cannot avoid doing so. Instead, he wants to color them such that the number of pairs of circles connected by a line segment and having the same color is minimal.

Help Baby Bob by writing a program that finds an optimal coloring of the circles.

☞ Among the attachments of this task you may find a template file `circlecoloring.*` with a sample incomplete implementation.

Input

The input file consists of:

- a line containing integers N and M, the number of circles and the number of extra line segments, respectively.
- M lines, the i-th of which consisting of integers U_i and V_i, meaning that Baby Bob drew a line segment connecting circle U_i and circle V_i.

Note that the original N line segments are not included in the input.

Output

The output file must contain two lines:

- a line containing a single integer K, the minimum number of connected pairs of circles that have the same color in an optimal coloring.
- a line consisting of the N integers $C_0, \ldots, C_{N-1}$, the description of an optimal coloring. C_i is 0 if circle i should be colored red, and 1, if it should be colored blue.

If there are multiple optimal colorings, you may output any of them.

Constraints

- $3 \leq N \leq 300\,000$.
- $0 \leq M \leq 100\,000$.
- $0 \leq U_i < V_i \leq N - 1$ for each $i = 0 \ldots M - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (4 points)	$M = 0$.
– **Subtask 3** (11 points)	$N \leq 20$.
– **Subtask 4** (8 points)	$M \leq 1$.
– **Subtask 5** (20 points)	$U_0 < U_1 < \ldots < U_{M-1} < V_{M-1} < V_{M-2} < \ldots < V_0$.
– **Subtask 6** (25 points)	$N \leq 3000$.
– **Subtask 7** (32 points)	No additional limitations.

In each subtask you can obtain a partial score by correctly determining the number of pairs of connected circles having the same color in an optimal coloring.

Specifically, you get 50% of the points of a subtask if the value of K is correct in each testcase of the subtask, but there is at least one testcase where the coloring given in the second line of the output contains more than K connected pairs with the same color. Note that the second line must contain a valid coloring, that is, N numbers with value 0 or 1, otherwise you may receive an “Output isn’t correct” verdict.

Examples

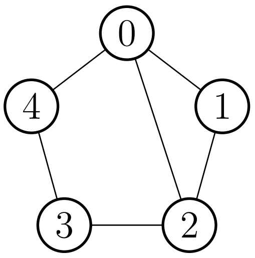
input	output

`5 1`
`0 2`
	`1`
	`0 0 1 0 1`

`9 3`
`1 8`
`2 4`
`5 7`
	`3`
	`0 1 0 1 0 1 1 0 1`

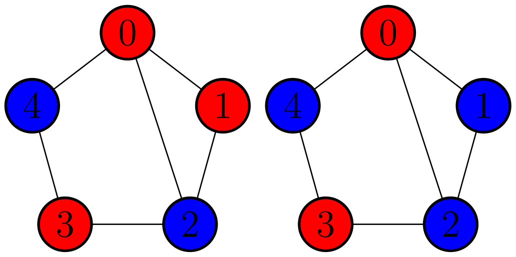
Explanation

In the **first sample case**, the image with the extra line connecting circles 0 and 2 is displayed in the following figure.



It can be proved that it is not possible to color the circles such that every connected pair has different colors. For example, at least two out of circles 0, 1 and 2 will always have the same color.

However, it is possible to color the circles such that only one connected pair has the same color. There are exactly four such colorings, two of them are displayed in the figure below.



Esoteric Bovines (esotericbovines)

Farmer Richard has a herd of $2 \cdot N$ mystical cows, meticulously divided between two barns. Every cow produces mystical milk which has a magic value between 1 and $2^{60} - 1$, inclusive.

Figure 1: Richard's barns are full of magic.

Unfortunately for Richard, his wife, Dara got sick and she requires a special type of *magic* milk to be cured. The *magic* milk is obtained by combining two kinds of mystical milk from two cows, **one from the first barn and one from the second barn**. The *magic* value of the combined milk is the **bitwise xor** of the magic values of the two kinds of milk.

In a rather bizarre turn of events, Richard had a dream where Șogard, the magical fairy responsible for the mystical properties of his cows, delivered a message. Șogard instructed Richard that if he wanted Dara to recover, she must consume the milk with the K-th smallest magic value out of all possible combinations of mystical milk.

In a hurry, Richard jumped from his bed and went to his barns, but unfortunately, his mystical cows had gone to the magic land to eat magic grass to regain their mystical strength. In his dizziness, he noted down T possible scenarios, each consisting of the magic value K given by Șogard, and two lists $A_0, A_1, \ldots, A_{N-1}$ and $B_0, B_1, \ldots, B_{N-1}$, the magic values of milk produced by the cows in the first barn and in the second barn.

Being out of touch with programming since he embraced the tranquil life of farming, Richard turns to you for help in finding the answers to his T scenarios.

☞ Among the attachments of this task you may find a template file `esotericbovines.*` with a sample incomplete implementation.

Input

The first line contains the only integer T, the number of scenarios Richard wrote.

The first line of each scenario contains N and K, the number of cows in each barn, and the position of the magic milk value he needs from the sorted list of all possible combinations.

The second line contains N integers $A_0, A_1, A_2, \ldots, A_{N-1}$, denoting the magic values of the milk produced in the first barn.

The third line contains N integers $B_0, B_1, B_2, \ldots, B_{N-1}$, denoting the magic values of the milk produced in the second barn.

Output

You need to write T lines, each containing one integer: the K-th smallest magic value that can be obtained by mixing two kinds of milk, one from the first barn and one from the second barn.

Constraints

- $1 \leq T \leq 10\,000$
- $1 \leq N \leq 200\,000$
- $1 \leq K \leq N^2$
- $1 \leq A_i < 2^{60}$ for each $i = 0 \ldots N-1$.
- $1 \leq B_i < 2^{60}$ for each $i = 0 \ldots N-1$.
- Additional constraint on the input: the sum of N over all scenarios doesn't exceed $200\,000$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (10 points)	$N \leq 1000$ and the sum of N over all scenarios doesn't exceed 1000.
– **Subtask 3** (15 points)	$A_i, B_i < 2^{20}$ and $K = 1$ or $K = N^2$.
– **Subtask 4** (30 points)	$N \leq 50\,000$; $A_i, B_i < 2^{20}$ and the sum of N over all scenarios doesn't exceed $50\,000$.
– **Subtask 5** (45 points)	No additional limitations.

Examples

input	output
`3`	`4`
`5 5`	`8`
`3 6 20 5 9`	`26`
`12 3 19 7 13`	
`5 10`	
`3 6 20 5 9`	
`12 3 19 7 13`	
`5 25`	
`3 6 20 5 9`	
`12 3 19 7 13`	

Explanation

In the **sample case**, the magic values in ascending order in each scenario are:

$$0, 1, 2, 4, 4, 5, 5, 6, 7, 8, 9, 10, 10, 11, 14, 14, 15, 16, 19, 21, 22, 23, 24, 25, 26.$$

So the **fifth** smallest magic value is 4, the **tenth** is 8 and the **twenty fifth** is 26.

Geometric Mean (geometricmean)

Alice found N cards. On each card there is a number written. While she is waiting for her friends to play some game, she decided to count how many ways can we choose 4 cards, such that the *geometric mean*[1]of the numbers on the cards is an integer number.

Can you help her to count the number of quadruples when the geometric mean is integer?

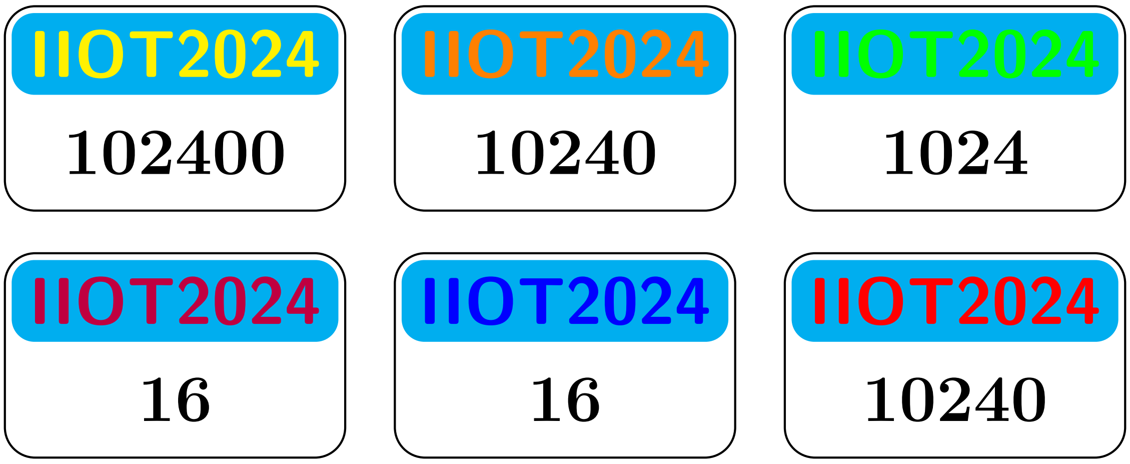


Figure 1: Six of the cards: $16, 16, 1024, 10240, 10240, 102400$.

Formally, given an array V of length N. We van to count the quadruples $0 \le i < j < k < \ell \le N-1$ such that $\sqrt[4]{V[i] \cdot V[j] \cdot V[k] \cdot V[\ell]}$ is an integer.

☞ Among the attachments of this task you may find a template file `geometricmean.*` with a sample incomplete implementation.

Input

The input file consists of:

- a line containing integer N: the number of the cards.
- a line containing the N integers $V_0, \ldots, V_{N-1}$: the numbers written on the cards.

Output

The output file must contain a single line consisting of a 64-bit integer: the number of quadruples of cards with integer geometric mean.

Constraints

- $1 \le N \le 1000$.
- $1 \le V_i \le 1\,000\,000$ for each $i = 0 \ldots N-1$.

[1]The geometric mean of a, b, c, d is $\sqrt[4]{abcd}$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
- **Subtask 2** (10 points) $N \leq 50$, and $V_i \leq 10^4$ for each $i = 0 \ldots N-1$.
- **Subtask 3** (10 points) $N \leq 50$.
- **Subtask 4** (10 points) V_i is a power of two for each $i = 0 \ldots N-1$.
- **Subtask 5** (25 points) $N \leq 200$.
- **Subtask 6** (45 points) No additional limitations.

Examples

input	output

`8`
`1 6 4 5 7 10 15 14`
	`0`

`7`
`2024 2024 2024 2024 2024 2024 2024`
	`35`

`8`
`1 2 4 8 16 32 64 128`
	`18`

Explanation

In the **first sample case** we can not choose 4 numbers, such that their geometric mean is integer.

In the **second sample case** any four-tuple is good, so the answer is $\binom{7}{4} = 35$.

In the **third sample case** we have 18 good quadruples. For example $(1, 2, 4, 32)$ is good because $\sqrt[4]{1 \cdot 2 \cdot 4 \cdot 32} = 4$. We list the rest of the good quadruples: $(1, 2, 8, 16)$, $(1, 2, 16, 128)$, $(1, 2, 32, 64)$, $(1, 4, 8, 128)$, $(1, 4, 16, 64)$, $(1, 8, 16, 32)$, $(1, 8, 64, 128)$, $(1, 16, 32, 128)$, $(2, 4, 8, 64)$, $(2, 4, 16, 32)$, $(2, 4, 64, 128)$, $(2, 8, 32, 128)$, $(2, 16, 32, 64)$, $(4, 8, 16, 128)$, $(4, 8, 32, 64)$, $(4, 32, 64, 128)$, $(8, 16, 64, 128)$.

Carlo's Garden (grasshopper)

Carlo hates bugs[2], he wants to kill every single one of them[3]. After months of hard work, he successfully killed all bugs in his garden, but when he was about to celebrate, a grasshopper jumped out of nowhere and started to annoy him.

Figure 1: Carlo killing the bugs.

Carlo's garden can be represented as a square grid of length 10^9. Initially, the grasshopper is standing at the point $(0, 0)$. The grasshopper can only jump by one cell to the East or to the North, i.e. it can jump from (x, y) to $(x + 1, y)$ or $(x, y + 1)$ but not to $(x - 1, y)$ or $(x, y + 2)$. Every seconds the grasshopper makes N jumps. For example if $N = 3$, after one second the grasshopper can be at $(0, 3)$, $(2, 1)$ but not at $(3, 3)$ or $(1, 1)$.

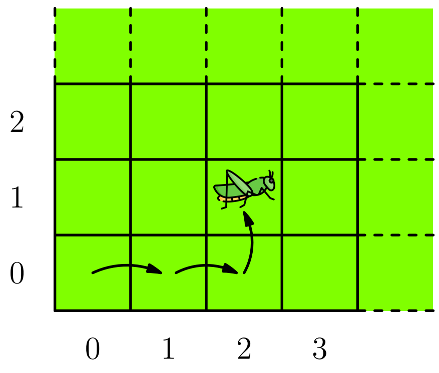


Figure 2: The N=3 grasshopper can arrive to $(2, 1)$ by for example first jumping to $(1, 0)$ and then to $(2, 0)$. In this case if there were any traps on $(1, 0), (2, 0)$ or $(2, 1)$ it would be caught.

Carlo wants to catch the grasshopper by placing traps in the garden, if the grasshopper jumps to a point where there is a trap, it will be caught and Carlo can finally celebrate. Unfortunately he can only place **one trap per second**.

Can you help Carlo to catch the grasshopper with at most 500 traps?

[2]The "animals" bugs, not the "informatics" bugs. He also hates informatics bugs by the way.
[3]Except bees, bees are cool.

☞ This is an **interactive** task! You have to communicate with another program using standard input and output. This means that you cannot simply read all the input file immediately, but what you read depends on what you previously wrote.

☞ Among the attachments of this task you may find a template file `grasshopper.*` with a sample incomplete implementation.

Interaction

Initially, you should read a single line containing the integer N.

For each round you should output a single line containing two integers B_x and B_y (which is not the grasshopper current position), both between 0 and 10^9 and the interactor responds with a single line containing either:

- "`-1 -1`" if the grasshopper was caught in that round;
- otherwise two integers G_x and G_y, the current position of the grasshopper.

If your output is incorrectly formatted, or the same as the grasshopper current position, or it has with an out of bounds value, or if you exceeded the number of traps, the interactor will send a single line containing "`-1 -1`".

After you receive "`-1 -1`", you must not output anything else, and you should terminate immediately.

Constraints

- $1 \le N \le 5$.
- Carlo's garden is a square grid $10^9 \times 10^9$.
- You can place at most 500 traps.
- The interactor is adaptive, meaning that it will change its behaviour based on your output.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (10 points) $N \le 1$.

– **Subtask 2** (20 points) $N \le 2$.

– **Subtask 3** (20 points) $N \le 3$.

– **Subtask 4** (20 points) $N \le 4$.

– **Subtask 5** (30 points) $N \le 5$.

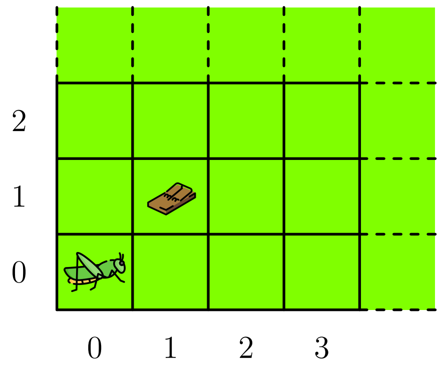
Examples

☞ This is an **interactive** task! What you read depends on what you've written, so there is not a unique interaction. The one presented here is just an example.

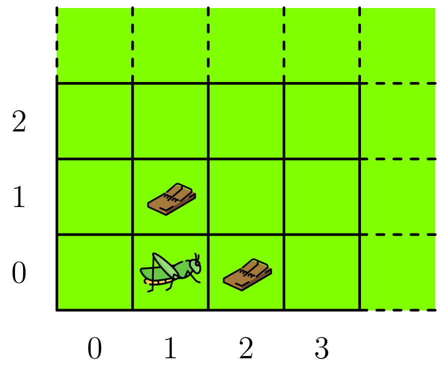
input	output
`1`	
	`1 1`
`1 0`	
	`2 0`
`-1 -1`	

In this example $N = 1$ so the grasshopper can only jump by one cell at a time.

Initially the grasshopper is at $(0, 0)$, Carlo places a trap at $(1, 1)$.



Then the grasshopper jumps to $(1, 0)$, Carlo places a trap at $(2, 0)$.



The grasshopper can only jump to $(2, 0)$ or $(1, 1)$, in both cases it will jump on a trap and Carlo can celebrate.

Replace the Characters (`replacechar`)

Given a string s of length n consisting of lowercase English letters, and let s_i be the i-th character of the string s.

You are allowed to modify the string using the following operation:

- Change one character to another. More formally choose an integer i ($1 \le i \le n$) and a lowercase English letter c, and set s_i to c.

What is the minimum number of operations that you need to do in order to transform s into a string that is sorted in non-decreasing order.

Also, print the sequence of operation that should be done in order to do so. (**Please refer to the output section for details on how to print the operations**).

If there are multiple answers, please output the one that **makes the resulting string lexicographically minimum**.

☞ Among the attachments of this task you may find a template file `replacechar.*` with a sample incomplete implementation.

Input

The first line of input contains a single integer n.

The second line of input contains a string s of length n consisting of lowercase English letters.

Output

The first line of output must contain a single integer k, the minimum number of operations.

The following k lines of output, each must contain an integer i ($1 \le i \le n$) and lowercase English letter c separated by a space, describing the operation.

Please note that if there are multiple answers, you should output the one that makes the resulting string lexicographically minimum.

Constraints

- $1 \le n \le 100\,000$
- $|s| = n$
- s_i is a lowercase English letter.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (15 points) $N \leq 10$.

– **Subtask 3** (30 points) $N \leq 1000$.

– **Subtask 4** (55 points) No additional limitations.

Examples

input	output

`2`
`ba`
	`1`
	`1 a`

`2`
`ab`
	`0`

Explanation

In the **first sample**, If we replace the first character with the letter 'a', then the string becomes "aa" which is sorted in non-decreasing order.

In the **second sample**, the string is already sorted in non-decreasing order, and no operations are needed.

Triangle Counting (`triangles`)

You are given a complete directed graph (a tournament graph) where all edges are initially directed towards the vertex with the higher number on it.

A triangle is a triple (a, b, c), such that we have the edges $a \to b, b \to c, c \to a$, with $(1 \leq a, b, c \leq n), a \neq b, a \neq c, b \neq c$.

We need to count how many triangles exist in the graph.

Since this is way too easy, you need to process q updates, where in each update we flip an edge and we are required to count the number of triangles in the given graph after all updates were processed up to that point.

☞ Among the attachments of this task you may find a template file `triangles.*` with a sample incomplete implementation.

Input

The first line of the input contains n and q, representing the number of vertexes in the graph and the number of updates.

The next q lines contain the updates of form (x, y), with $1 \leq x, y \leq n, x \neq y$. It is guaranteed that there is an edge between x and y in the given graph, directed towards y.

Output

The output will contain q lines, namely the answer after each update was processed.

Constraints

- $1 \leq n \leq 200\ 000$.
- $1 \leq q \leq 10^6$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (14 points)	$n, q \leq 100$.
– **Subtask 3** (16 points)	$n \leq 100$.
– **Subtask 4** (23 points)	$n \leq 2\ 000$.
– **Subtask 5** (47 points)	No additional limitations.

Examples

input	output
4 4	1
2 4	2
1 2	0
4 2	1
2 3	

Explanation

In the **first sample case**.

TV Series (tvseries)

Alessandro is a big fan of TV series and is eager to discuss them with his friends. In the next D days, they will be talking about N different series, numbered from 0 to $N-1$, with series i being discussed from day S_i to day E_i included. Note that time passes fast when having fun, thus in a single day they will discuss **at most one** series.

Figure 1: Alessandro cannot afford to miss too many discussions!

Alessandro has not watched any of those series yet, but he wants to talk with his friends as much as possible. However he only has a limited amount of free time and knows it will take him X_i days to watch series i. Alessandro can watch the series in any order and skip some, but he cannot start watching a new series before finishing the previous one. In order to take part in the discussion on day d, Alessandro must have watched completely the tv series which is being talked about by the end of day $d-1$.

For how many days, at most, can he take part in the discussion with his friends?

☞ Among the attachments of this task you may find a template file `tvseries.*` with a sample incomplete implementation.

Input

The input file consists of:

- a line containing integers N, D: the number of TV series that will be discussed and the number of days.
- a line containing the N integers $S_0, \ldots, S_{N-1}$.
- a line containing the N integers $E_0, \ldots, E_{N-1}$.
- a line containing the N integers $X_0, \ldots, X_{N-1}$.

Output

The output file must contain a single line consisting of integer K, the maximum number of days in which Alessandro can discuss TV series with his friends.

Constraints

- $1 \leq N \leq 2000$.
- $1 \leq D \leq 5000$.
- $1 \leq S_i, E_i, X_i \leq D$ for each $i = 0, 1, \ldots N-1$.
- $S_i \leq E_i \leq S_{i+1}$ for each $i = 0, 1, \ldots N-2$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (18 points) $N \leq 18$, $D \leq 100$

– **Subtask 3** (25 points) $N \leq 100$, $D \leq 100$.

– **Subtask 4** (57 points) No additional limitations.

Examples

input	output

`2 4`
`2 4`
`2 4`
`1 2`
	`2`

`4 10`
`2 3 7 8`
`2 6 7 10`
`1 4 3 2`
	`5`

Explanation

In the **first sample case** there are 2 TV series and 4 days.

- Series 0 will be discussed on day 2 and takes 1 day to watch.
- Series 1 will be discussed on days 3 and 4 and takes 2 days to watch.

It is optimal for Alessandro to watch series 1 on day 1 and discuss it on day 2, then watch series 2 on days 2 and 3 and discuss it on day 4. This will let Alessandro take part in both days of discussion.

In the **second sample case** there are 4 TV series and 10 days.

- Series 0 will be discussed on day 2 and takes 1 day to watch.
- Series 1 will be discussed on days 3, 4, 5 and 6 and takes 4 days to watch.
- Series 2 will be discussed on day 7 and takes 3 days to watch.
- Series 3 will be discussed on days 8, 9 and 10 and takes 2 days to watch.

It is optimal for Alessandro to watch series 1 on days 1, 2, 3 and 4, and discuss it on days 5 and 6 and to watch series 3 on days 6 and 7, then discuss it on days 8, 9 and 10. This will let Alessandro participate to the discussion on 5 days, and it can be proven that he cannot do better.

Washington Distance (washington)

Ethan and George are in Washington D.C. for their favorite debate tournament's final round. To get to the venue, they need to use a taxi, and the driver expects the directions given according to the city's unique street system described below. The starting point and destination is an intersection of two streets.

The streets of Washington form a rectangular grid, which can be modeled as vertical and horizontal lines in the planar coordinate system. There is a street parallel to the Ox and Oy axis on each integer coordinate from −25 to 25. However, the streets are numbered in a very strange way.

Firstly, the city is divided into four quadrants: NW, SW, NE, SE (north-west, south-west, north-east, south-east), relative to the center of the city (which corresponds to the origin in the coordinate system). The vertical streets are identified by letters from A to Z. Street A goes through the center, and on both the eastern and western sides the streets are named B, C, ..., Z in the order of distance from the center. The horizontal streets are identified by numbers between 0 and 25. Street 0 goes through the center, and in both the northern and southern halves, the streets are numbered from the bottom to the top with increasing numbers from 1 to 25 (so the numbers reflect the remainder of the real coordinate when divided by 26). See the image below for a better understanding.

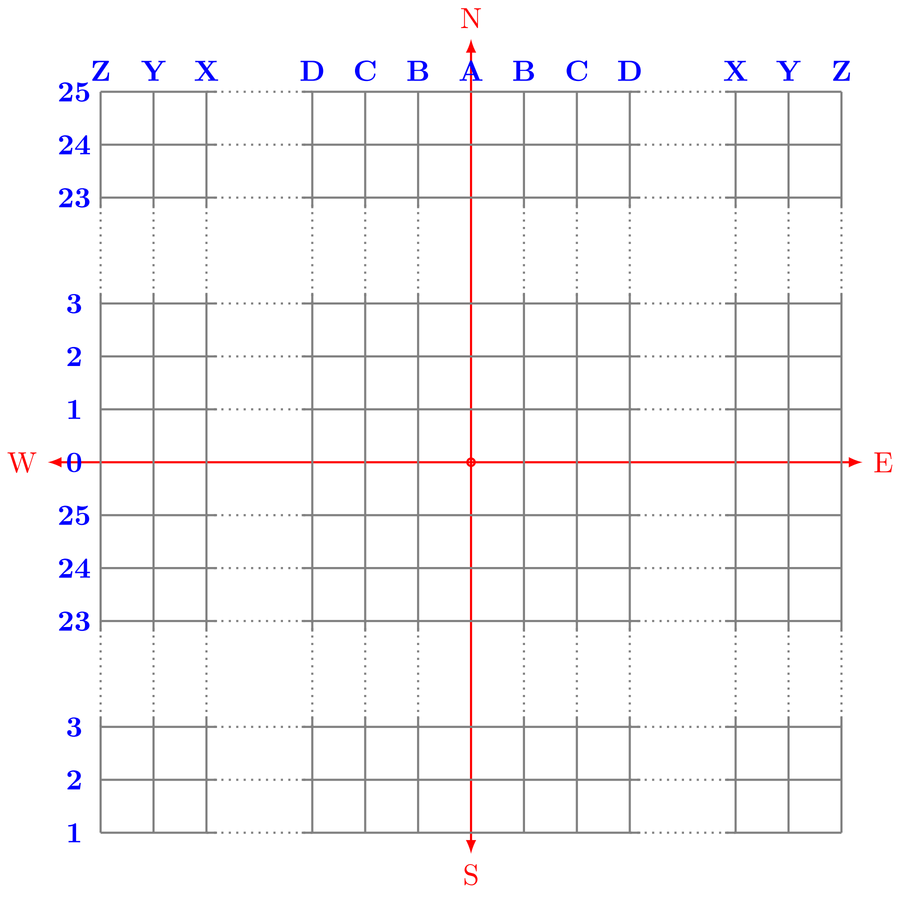


Figure 1: Map of Washington D.C.

Now, knowing this, Ethan and George want to know the distance between T pairs of points, according to the Washington Coordinate system. The distance means the length of the ride with the taxi, which can only travel along the streets, given in units of the coordinate system.

☞ Among the attachments of this task you may find a template file `washington.*` with a sample incomplete implementation.

Input

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

The next T lines of the input contain two addresses according to the Washington Coordinate System, given by their direction, letter and street number.

Output

You need to write a single line with an integer: the unique integer that solves this task.

Constraints

- $1 \leq T \leq 10^4$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (20 points) All addresses are in the NW quadrant.

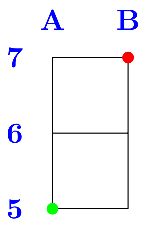
– **Subtask 3** (80 points) No additional limitations.

Examples

input	output
5 NE A 5 NE B 7 NE G 15 SW P 0 SE U 5 NW Q 10 NW A 19 SW B 3 SE Q 21 SW P 4	3 36 67 43 48

Explanation

The situation of the **first sample case** can be seen below: (clearly the distance is 3)



In the **second sample case** the taxi driver does not have to cross line 0, but must cross line A. The distance between line G (corresponding to the starting point) and line A is 6. The distance between line P (corresponding to the end point) and line A is 15. So the total distance is $6 + 15 + 15 = 36$.

In the **third sample case** the taxi driver must cross both line 0 and line A. The distance between line U (corresponding to the starting point) and line A is 20. The distance between line Q (corresponding to the end point) and line A is 16. So the total distance is $20 + 16 + 21 + 10 = 67$ as the distance between line 5 (corresponding to the starting point) and line 0 is 21.

Watch Towers (`watchtowers`)

You are a builder in the kingdom, and to protect it, the king has ordered to construct N watchtowers each of height H_i, where watchtower i is installed at $x = i$.

After constructing the towers, You have realised that there are watchtowers that can't see other towers because there are other towers blocking the view.

Watchtower i can see watchtower j if you can draw a straight line from tower i to tower j without it intersecting with the body of any other tower (but it can intersect with the top of another tower).

As this kingdom is a magical kingdom, You can use magic to increase the height of any tower you like by an integer $h >= 0$, but you can only use it once, and because using magic is hard, you want to increase it by the minimum height such that the tower you chose now can see every other tower.

The king is going to do a suprise checkup visit to one of the towers, so you hace to determine for each tower i the minimum height h to be increased such that you can see every other tower from the top of it.

It can be proven, that under the given conditions, the answer of each tower is always an integer.

☞ Among the attachments of this task you may find a template file `watchtowers.*` with a sample incomplete implementation.

Input

The input file consists of:

- a line containing integer N.
- a line containing the N integers $H_0, \ldots, H_{N-1}$.

Output

The output file must contain a single line consisting of the N integers $D_0, \ldots, D_{N-1}$.

Constraints

- $1 \leq N \leq 200\,000$.
- $1 \leq H_i \leq 1\,000\,000\,000$ for each $i = 0 \ldots N-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (20 points) $N \leq 100$.

– **Subtask 3** (31 points) $N \leq 5000$.

– **Subtask 4** (49 points) No additional limitations.

Examples

input	output

`5`
`4 3 1 3 4`
	`1 0 1 0 1`

`2`
`3 7`
	`0 0`

Explanation

In the **first sample case**, tower $i = 1$ cannot see tower $j = 3$ because tower $k = 2$ is blocking the view, so increaseing i by $h = 1$ would make it visible, and same goes for $i = 3$ and $i = 5$.

In the **second sample case**, both towers can see eachother, so no need to increase either of them.