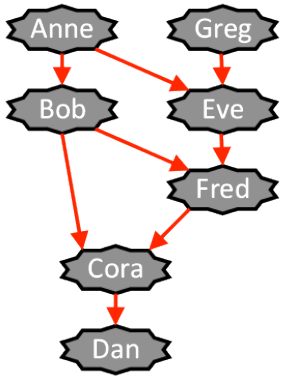
7272 Promotions

The Fair Inc. administration decided to promote the best employees and limited the number of promotions to a fixed interval $[A, B]$. The directors compared the employees' performance and their evaluations resulted in a consistent precedence relation among employees, which has to be respected by promotions. This means that, for every pair of employees x and y, if x outperformed y, then y may be promoted only if x is promoted.

In order to understand whether the data collected so far is enough for ensuring fairness, the executive chairman wants to know:

- How many employees will certainly be promoted in the interval endpoints (i.e., if the number of promotions is A and if the number of promotions is B).
- How many employees have no possibility of being promoted (even if the number of promotions is B).



Consider the example depicted in the figure. There are seven employees and eight precedence rules. An arrow from an employee x to an employee y means that x outperformed y. The number of promotions is limited to the interval $[3, 4]$. Therefore:

- If there are only three promotions, the promoted employees must be:
  - either Anne, Bob and Greg,
  - or Anne, Eve and Greg.

  In this case, two employees (Anne and Greg) will certainly be promoted. Notice that, with the current information, Bob and Eve may or may not win a promotion.
- If there are four promotions, the promoted employees have to be:
  - Anne, Bob, Eve and Greg.

  So, with four promotions, four employees (Anne, Bob, Eve and Greg) will certainly be promoted and three employees (Cora, Dan and Fred) have no possibility of being promoted.

Write a program that, given the interval of the number of promotions, the set of employees and the precedence relation among them, computes, for each of the interval endpoints, the number of employees that will certainly be promoted, and the number of employees that have no possibility of being promoted.

The precedence relation is consistent in the sense that, if an employee x outperformed an employee y, y did not outperform (directly or indirectly) x.

Input

The input file contains several test cases, each of them as described below.

The first line of the input has four space separated integers: A, B, E, and P. A and B are the interval endpoints, E is the number of employees and P is the number of precedence rules. Employees are identified by integers, ranging from 0 to $E - 1$.

Each of the following P lines contains two distinct space separated integers, x and y, which indicate that employee x outperformed employee y.

Constraints:

$1 \leq A < B < E$	Interval endpoints.
$2 \leq E \leq 5\,000$	Number of employees.
$1 \leq P \leq 20\,000$	Number of precedence rules.

Output

For each test case, the output consists of three lines. The first line contains the number of employees that will certainly be promoted if there are A promotions. The second line contains the number of employees that will certainly be promoted if there are B promotions. The third line contains the number of employees that have no possibility of being promoted (even if there are B promotions).

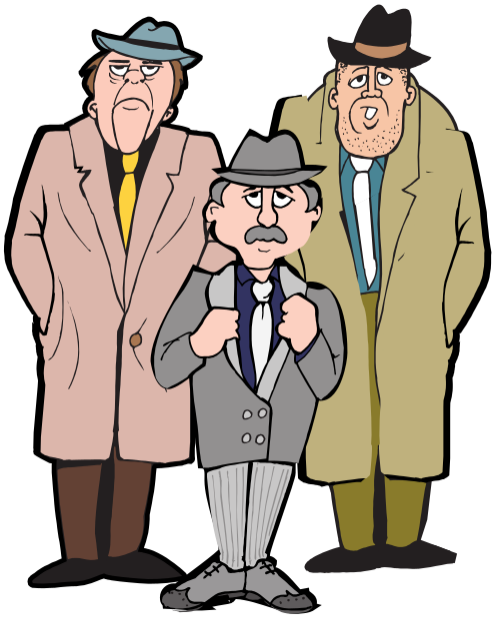
Sample Input

```
3 4 7 8
0 4
1 2
1 5
5 2
6 4
0 1
2 3
4 5
```

Sample Output

```
2
4
3
```

7273 Black Vienna



Black Vienna is a puzzle game where players try to deduce the secret identities of the three spies. There are 26 suspects, represented by cards with a single letter from 'A' to 'Z'. Suspect cards are shuffled and three are secretly set aside; these form the *Black Vienna* circle. The remaining 23 cards are distributed between the two players. Note that players will necessarily get a different number of cards; one player may also get all 23 cards while the other gets none.

The objective of the puzzle is to deduce which of the suspects are in the Black Vienna circle using player's replies to *investigations*; each investigation consists of a pair of suspects and the player's reply is the number of those suspects that are in his/her hand. Using several investigations it is possible to narrow which suspects can be in the Black Vienna circle (i.e., those that are *not* in any of the player's hands).

Write a program that reads a sequence of investigation replies and counts the *number of admissible solutions*, i.e. possible sets of three suspects representing the members of the Black Vienna circle. Note that it is possible that the player's replies are inconsistent and therefore that the puzzle has no solution.

Input

The input file contains several test cases, each of them as described below.

The input consists of a line with the number N of investigations followed by N lines; each line consists of a sequence of two distinct letters (from 'A' to 'Z'), a player number (1 or 2) and a reply (an integer from 0 to 2).

Constraints:

$0 \leq N \leq 50$ Number of investigations.

Output

For each test case, the output is the number of distinct admissible solutions, i.e. sets of three members of the Black Vienna circle.

Sample Input

```
0
3
AB 1 1
AC 2 1
BC 2 1
3
AB 1 2
```

```
AC 2 1
BC 1 0
```

Sample Output

```
2600
506
0
```

7274 Canvas Painting

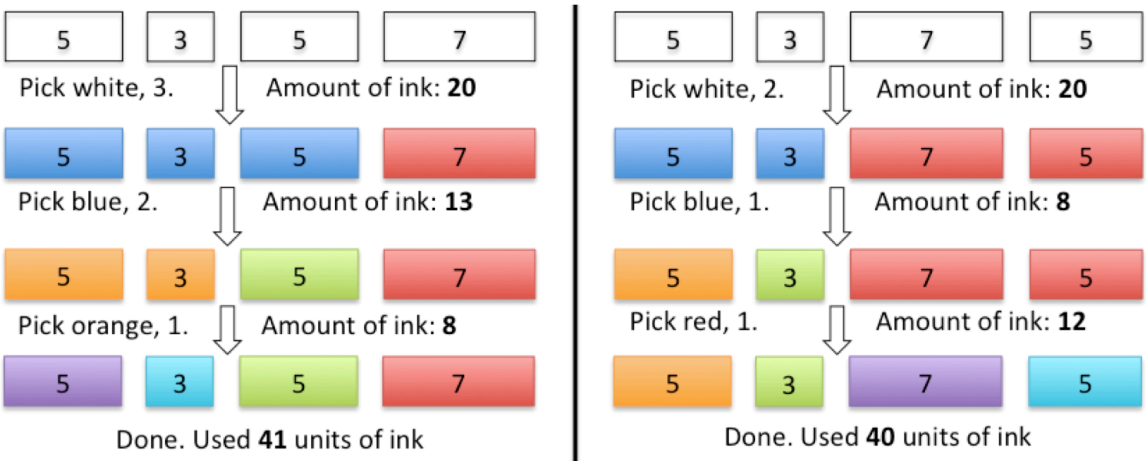
After last year's success, Samuel W. E. R. Craft's fame continues to grow and now he has funds for all kinds of projects that cross his mind. His newest idea involves creating arrays of canvasses with color patterns having no repeated colors.

Samuel bought a set of white canvasses of varying sizes. Since painting them manually would take too much time, he devised a huge machine to automate the painting process.

The painting process works as follows:

1. Assemble all canvasses in a line in the machine's conveyor belt, disposed in some chosen order.
2. Pick a color C and a number F (which should be less than the number of color C canvasses).
3. Going from left to right, all canvasses with color C are painted. The first F color C canvasses are painted with a new color X and the remaining color C canvasses are painted with a new color Y. Colors X and Y are selected by the machine, are distinct, and are different from any color used previously. The amount of ink spent in this step is equal to the sum of the sizes of the painted canvasses.
4. Repeat 2) and 3) until all canvasses have distinct colors.

Consider for example that Samuel bought four canvasses of sizes 3, 5, 5 and 7. The following picture shows 2 different options for painting them.



Given the sizes of the canvasses Samuel bought, can you help Samuel finding the minimum amount of ink the machine needs to spend in order to have all canvasses with different colors?

Input

The first line consists of a single integer T, the number of test cases. Each test case is composed by two lines. The first line consists of a single integer N representing the number of canvasses. The next line contains N space separated integers representing the sizes of the canvasses.

Constraints:

$1 \leq T \leq 100$	Number of test cases.
$1 \leq N_i \leq 100\,000$	Number of canvasses in the i^{th} test case.
$1 \leq s \leq 100\,000$	Size of each canvas.
$1 \leq \sum_{i=1}^{T} N_i \leq 100\,000$	Number of canvasses over all test cases in one test file.

Output

The output contains T lines, one for each test case: the minimum amount of ink the machine needs in order to have all canvasses with different colors.

Sample Input

```
2
3
7 4 7
4
5 3 7 5
```

Sample Output

```
29
40
```

7275 Dice Cup

In many table-top games it is common to use different dice to simulate random events. A "*d*" or "*D*" is used to indicate a die with a specific number of faces, *d4* indicating a four-sided die, for example. If several dice of the same type are to be rolled, this is indicated by a leading number specifying the number of dice. Hence, *2d6* means the player should roll two six-sided dice and sum the result face values.

Write a program to compute the most likely outcomes for the sum of two dice rolls. Assume each die has numbered faces starting at 1 and that each face has equal roll probability.

Input

The input file contains several test cases, each of them as described below.

The input consists of a single line with two integer numbers, N, M, specifying the number of faces of the two dice.

Constraints:

$4 \leq N, M \leq 20$ Number of faces.

Output

For each test case, a line with the most likely outcome for the sum; in case of several outcomes with the same probability, they must be listed from lowest to highest value in separate lines.

The outputs of two consecutive cases will be separated by a blank line.

Sample Input

```
6 6
6 4
12 20
```

Sample Output

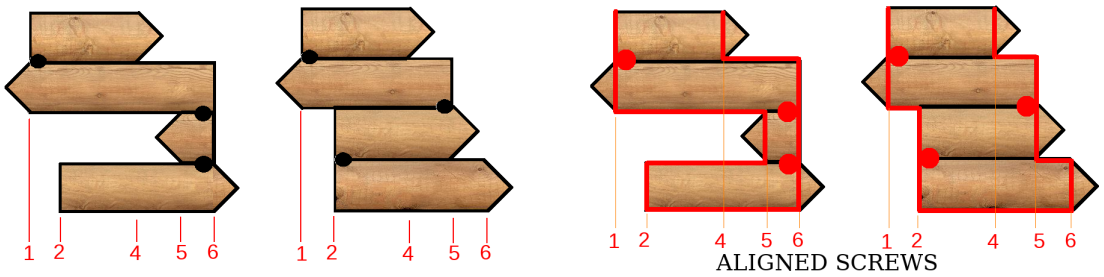
```
7

5
6
7

13
14
```

15
16
17
18
19
20
21

7276 Wooden Signs

A carpenter has received an order for a wooden directional sign. Each board must be aligned vertically with the previous one, either to the basis of the previous arrowhead or to the opposite side, being fixed there with a specially designed screw. The two boards must overlap.

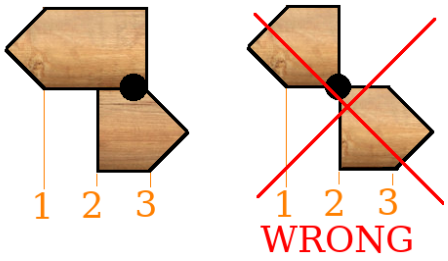
The carpenter wrote down a sequence of integers to encode the sketch sent by the designer but the sequence does not determine a unique model and he has thrown away the original sketch. What looked like a trivial task turned out a big jigsaw to him.



The sequence (with $1+N$ elements) encodes the (N) arrows from the bottom to the top of the sign. The first element is the position of the left side of the bottom arrow. The remaining N elements define the positions where the arrowheads start, from bottom to top: the i-th element is the position where the i-th arrowhead basis is. For instance, the two signs depicted (on the left and on the right) could be encoded by `2 6 5 1 4`.

Since a board must be aligned vertically with the previous one (either to the basis of the previous arrowhead or to the opposite side), if the sequence was `2 6 5 1 4 3`, the fifth arrow could be fixed (in any of the depicted signs) with a screw either at 1 (pointing to the right) or at 4 (pointing to the left), with the arrowhead basis at 3.

If the sequence was `2 3 1`, the second arrow could only be fixed with a screw at 3, pointing to the left, because consecutive boards must overlap.



All arrowheads are similar and the designer told the carpenter that their bases stand in different vertical lines, as well as the left side of the bottom arrow, altogether forming a permutation of $1..(N+1)$. That is why the carpenter overlooked the details and just wrote down the permutation (e.g., `2 6 5 1 4 3`).

Given the sequence of numbers the carpenter wrote down, compute the number of directional arrows signs that can be crafted. Since the number can be very large, you must write it modulo $2^{31}-1 = 2147483647$. The second integer in the sequence is always greater than the first one (the bottom arrow points to the right always).

Input

The input file contains several test cases, each of them as described below.

The first line has one integer N and the second line contains a permutation of the integers from 1 to $N+1$. Integers in the same line are separated by a single space.

Constraints:

$1 \leq N < 2000$ Number of Arrows.

Output

For each test case, the output has a single line with the number (**modulo** $2^{31} - 1 = 2147483647$) of distinct signs that can be described by the given permutation.

Sample Input

```
5
2 6 5 1 4 3
2
2 3 1
20
3 21 10 15 6 9 2 5 20 13 17 19 14 7 11 18 16 12 1 8 4
```

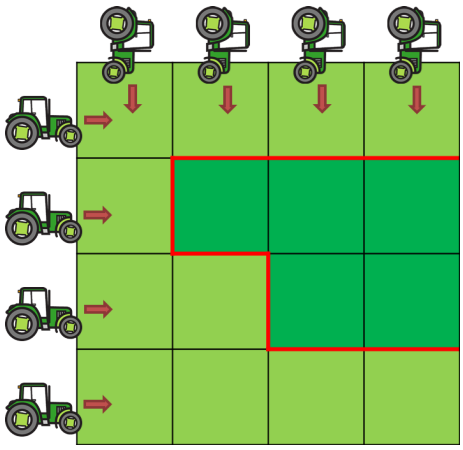
Sample Output

```
6
1
1887
```

7277 Landscaping

Preparations for a good harvest in Spring start now and farmer John is preparing his field for a good season. He went over-budget last year, as the tractors moving up and down the hills needed more fuel than he expected.

When harvesting, his tractors need to move both horizontally and vertically across all the land. In the image, you can see a region at low altitude in light green and a region at high altitude in dark green. When harvesting, his tractors will have to cross all the hills marked in red and they will have to go up or down 8 times.



This year, he is wondering whether he should level some parts of his field before sowing in order to lower his harvesting costs later on. Can you help him decide where the bulldozers should work in order to lower his costs?

Farmer John knows that his tractors need A additional euros when moving between adjacent patches of land at different heights. He can also pay B euros to either increase or decrease the height of any patch in his field.

What is the minimum amount of money he will have to pay this season?

Given a description of the field, the price to change the height of a patch of land and the price his tractors pay when moving between adjacent patches, the goal is to find out the minimum amount that farmer John will have to pay this year.

Input

The input file contains several test cases, each of them as described below.

The first line consists of 4 space separated integers, N, M, A and B. N and M represent the dimensions of his $N \times M$ field, A represents the cost to move between adjacent patches of land at different levels and B is the cost to change any patch of land.

The next N lines each have M characters and represent farmer John's field. A '.' signals a patch of land at a low level and a '#' represents a patch of land at a high level.

Constraints:

$1 \leq N, M \leq 50$	Size of the field.
$1 \leq A, B \leq 100\,000$	Cost to change any height or to move between adjacent patches.

Output

For each test case, you should output a single line with a single integer representing the minimum amount of money that farmer John will have to pay.

Sample explanation:

Farmer John has a 5×4 field. Moving between adjacent patches at a different level requires €1000 in fuel, while changing the height of a patch costs €2000.

Farmer John needs €11000: €2000 to change the isolated patch to a lower level and €9000 for the fuel needed to move between patches of land at different levels.

Not changing any patch would cost him €12000, changing all the high patches to low would cost him €18000, and changing all the low patches to high would cost him €22000.

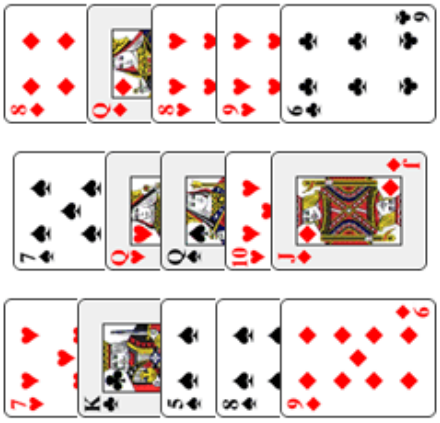
Sample Input

```
5 4 1000 2000
...#
#..#
...#
##..
###.
```

Sample Output

```
11000
```

7278 Game of Cards



Alice and Bob created a new game while at the beach this summer. All they need is a set of numbered playing cards. They start by creating P piles with all cards face-up and select a non-negative number K. After that, they take turns like this:

1. A player starts by selecting one of the piles.
2. Then, he removes from 0 up to K cards from the top of that pile, leaving at least one card in the pile.
3. Next, he looks at the card left at the top of the pile and must remove a number of cards equal to its value (from the top of the same pile).

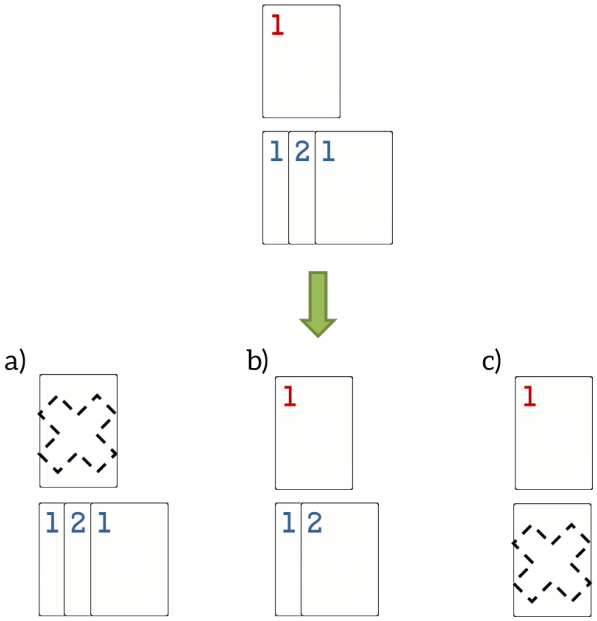
Whoever doesn't have more cards to remove, or whoever is forced to remove more cards than those available on a pile, loses the game.

In the figure, you can see an example with two piles and $K = 1$. The player to move might:

a) Select the first pile and 0 cards to remove, being forced to remove 1 card from the top next.

b) Select the second pile and 0 cards to remove, having to remove 1 card from the top next.

c) Select the second pile and 1 card to remove, having to remove 2 cards from the top next.



Alice has realized that Bob is very good at this game and will always win if he has the chance. Luckily, this time Alice is first to play. Is Alice able to win this game?

Given the description of the piles with all the cards and the maximum number of cards they can start to remove, your goal is to find out whether Alice can win the game if she is the first to play.

Input

The input file contains several test cases, each of them as described below.

The first line contains 2 space separated integers, P, the number of piles, and K, the maximum number of cards they can start to remove on their turn. The next P lines start with an integer N, indicating the number of cards on a pile. N space separated integers follow, representing the cards on that pile from the bottom to the top.

Constraints:

$1 \leq P \leq 100$	Number of piles.
$1 \leq K \leq 10$	Maximum number of cards a player can start to remove.
$1 \leq c \leq 10$	Number on each card.
$1 \leq N \leq 1\,000$	Size of each pile.

Output

For each test case, a single string, stating '`Alice can win.`' or '`Bob will win.`', as appropriate.

Notes:

Explanation of output 1. The piles are the same, so Bob will always be able to mirror whatever move Alice makes.

Explanation of output 2. Alice can start by removing 0 cards from the second pile and then 1 card from its top. Two legal moves will be possible next, Bob will make one and Alice the other.

Sample Input

```
4 1
4 1 1 1 1
6 2 1 2 1 2 1
4 1 1 1 1
6 2 1 2 1 2 1
2 1
1 1
3 1 2 1
2 2
5 3 2 1 2 1
5 5 4 3 2 1
```

Sample Output

```
Bob will win.
Alice can win.
Alice can win.
```

7279 Sheldon Numbers

According to Sheldon Cooper, the best number is 73. In his own words, *"The best number is 73. 73 is the 21st prime number. Its mirror, 37, is the 12th, and its mirror, 21, is the product of multiplying 7 and 3. In binary, 73 is a palindrome: 1001001, which backwards is 1001001. Exactly the same."*

Prime numbers are boring stuff, and so are palindromes. On the other hand, the binary representation of 73 is rather remarkable: it's 1 one followed by 2 zeroes, followed by 1 one, followed by 2 zeros, followed by 1 one. This is an interesting pattern that we can generalize: N ones, followed by M zeros, followed by N ones, followed by M zeros, etc, ending in either N ones or M zeroes. For 73, N is 1, M is 2, and there are 5 runs of equal symbols. With $N = 2$, $M = 1$ and 4 runs, we would have the string 110110, which is the binary representation of 54.

Acknowledging Sheldon's powerful insight, let us introduce the concept of a Sheldon number: a positive integer whose binary representation matches the pattern $ABABAB\ldots ABA$ or the pattern $ABABAB\ldots AB$, where all the occurrences of A represent a string with N occurrences of the bit 1 and where all the occurrences of B represent a string with M occurrences of the bit 0, with $N > 0$ and $M > 0$. Furthermore, in the representation, there must be at least one occurrence of the string A (but the number of occurrences of the string B may be zero).

Many important numbers are Sheldon numbers: 1755, the year of the great Lisbon earthquake, 1984, of Orwellian fame, and 2015, the current year! Also, 21, which Sheldon mentions, is a Sheldon number, and so is 42, the answer given by the Deep Thought computer to the Great Question of Life, the Universe and Everything.

Clearly, there is an infinite number of Sheldon numbers, but are they more dense or less dense than prime numbers?

Your task is to write a program that, given two positive integers, computes the number of Sheldon numbers that exist in the range defined by the given numbers.

Input

The input file contains several test cases, each of them as described below.

The input contains one line, with two space separated integer numbers, X and Y.

Constraints:

$0 \leq X \leq Y \leq 2^{63}$

Output

For each test case, the output contains one line, with one number, representing the number of Sheldon numbers that are greater or equal to X and less or equal to Y.

Notes:

Explanation of output 1. All numbers between 1 and 10 are Sheldon Numbers.

Explanation of output 2. 73 is the only Sheldon number in this range.

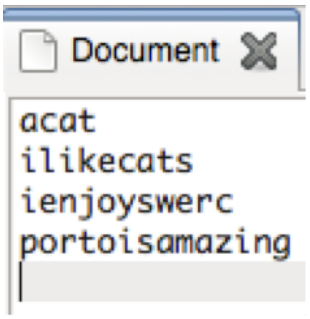
Sample Input

```
1 10
70 75
```

Sample Output

```
10
1
```

7280 Text Processor



Inês is new to programming. She is writing a simple application to handle text. So far, the app only supports adding text to a blank document. Rita, her restless little sister, is fascinated with all the different symbols on Inês' screen and wants to play with it, not letting Inês work any longer. In order to be able to continue working, Inês decides to turn it into a game for Rita.

Inês will let Rita write whatever she likes in the app. Then, Inês will select portions of the document with width W and challenge Rita to guess how many distinct sequences of symbols there are in that portion of text. Rita is excited to play, but there is a problem: Inês doesn't know yet how to write a program to find the right answer for her own questions. Can you help her out?

Given the description of the game being played by the two sisters: the text written in the document, and Inês' questions, find the number of distinct (non-empty) substrings for each of the questions. A substring is a sequence of contiguous letters in the document.

Input

The input file contains several test cases, each of them as described below.

The first line of the input contains the text written by Rita. The text is composed only by letters $[a - z]$. The second line has two space separated integers: Q and W. Q is the number of questions and W the fixed width of Inês' questions. Each of the following Q lines contains a single integer i, describing a question to know the number of distinct substrings in the range $[i, i + W - 1]$.

Constraints:

$1 \leq \|D\| \leq 100\,000$	Number of letters in a document.
$1 \leq Q \leq 100\,000$	Number of questions.
$1 \leq W \leq \|D\|$	Width of Inês' questions.
$1 \leq i \leq \|D\| - W + 1$	The question range is one-indexed and inside the document bounds.

Output

For each test case, the output contains the answer to each question in the same order as in the input, one per line.

Explanation of output 1:

The first question concerns the range [1, 3] → `aca`, which has 5 distinct substrings {`a`, `c`, `ac`, `ca`, `aca`}.

The second question corresponds to `cat`, which has 6 distinct substrings: {`a`, `c`, `t`, `ca`, `at`, `cat`}.

Sample Input

```
acat
2 3
1
2
portoisamazing
2 7
```

6
3

Sample Output

5
6
26
28

7281 Saint John Festival

Porto's Festa de São João is one of Europe's liveliest street festivals. Its peak is the night of 23rd to 24th of June, with dancing parties from Ribeira to Foz all night long.

Time to celebrate, with friends, relatives, neighbours or simply with other people in streets, armed with colored plastic hammers, huge garlic flowers or a bunch of lemongrass to gently greet passers-by. Fireworks, grilled sardines, barbecues, bonfires, potted basil plants (*manjericos*) and the sky covered by incandescent sky lanterns (*balões de S.João*) launched from every corner make this party unique.

The sky lanterns are made of thin paper and cannot be released until they are filled in with hot air. Sometimes they burn out still on ground or on the way up, if a sudden gust of wind catches them. For this reason, the successful launchers usually follow the movement of their sky lanterns, with a mixture of anxiety and joy, for as long as they can distinguish them in the sky.

We are not aware of any attempt to achieve a Guinness record of sky lanterns launched simultaneously (it could be dreadful night for firemen if there were).

Can you imagine, thousands of people preparing their sky lanterns for release at the city park, within a region of larger ones that will be launched simultaneously?

The large sky lanterns can be used to identify their positions in the sky afterwards, in order to count the surviving ones at an observation instant.

Given the positions of the large sky lanterns and the positions of the small ones, determine the number of small sky lanterns that are in the interior or on the boundary of some triangle defined by any three of the large ones.

Input

The input file contains several test cases, each of them as described below.

The first line has an integer L that defines the number of the large sky lanterns at the observation instant. Each of the following L lines contains a pair of integers separated by a space that gives the coordinates (x, y) of a large sky lantern. After that, there is a line with an integer S that defines the number of small sky lanterns and S lines, each defining the position of a small sky lantern. The height is irrelevant for us. All the given points are distinct and there are at least three points representing large sky lanterns that are not collinear.

Output

For each test case, the output has a single line with the number of small sky lanterns that are in the interior or on the boundary of some triangle defined by any three of the large sky lanterns.

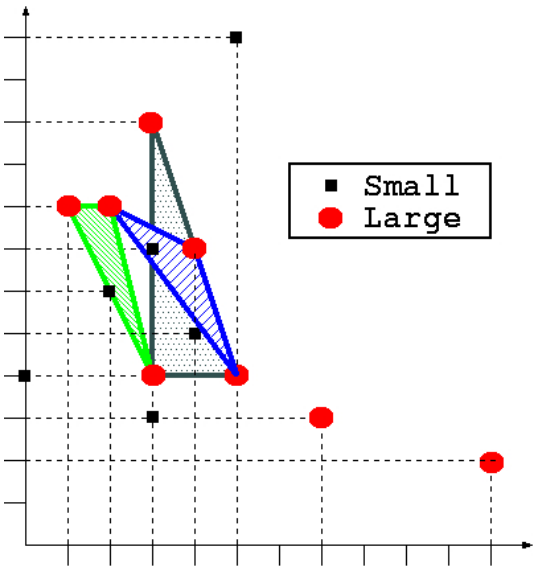
Constraints:

$3 \leq L \leq 10\,000$	Number of large sky lanterns.
$1 \leq S \leq 50\,000$	Number of small sky lanterns.
$0 \leq x, y \leq 2^{30}$	Bounds for coordinates.

Note: The picture on the right illustrates the sample input below

Sample Input

```
8
3 4
2 8
5 4
1 8
4 7
3 10
11 2
7 3
6
5 12
3 7
3 3
4 5
0 4
2 6
```



Sample Output

```
3
```