A. I_love_%username%

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya adores sport programming. He can't write programs but he loves to watch the contests' progress. Vasya even has a favorite coder and Vasya pays special attention to him.

One day Vasya decided to collect the results of all contests where his favorite coder participated and track the progress of his coolness. For each contest where this coder participated, he wrote out a single non-negative number — the number of points his favorite coder earned in the contest. Vasya wrote out the points for the contest in the order, in which the contests run (naturally, no two contests ran simultaneously).

Vasya considers a coder's performance in a contest *amazing* in two situations: he can break either his best or his worst performance record. First, it is amazing if during the contest the coder earns strictly **more** points that he earned on each past contest. Second, it is amazing if during the contest the coder earns strictly **less** points that he earned on each past contest. A coder's first contest isn't considered amazing. Now he wants to count the number of amazing performances the coder had throughout his whole history of participating in contests. But the list of earned points turned out long and Vasya can't code... That's why he asks you to help him.

Input

The first line contains the single integer n $(1 \leq n \leq 1000)$ — the number of contests where the coder participated.

The next line contains n space-separated non-negative integer numbers — they are the points which the coder has earned. The points are given in the chronological order. All points do not exceed 10000.

Output

Print the single number — the number of amazing performances the coder has had during his whole history of participating in the contests.

Sample test(s)

input

```
5
100 50 200 150 200
```

output

```
2
```

input

```
10
4664 6496 5814 7010 5762 5736 6944 4850 3698 7242
```

output

```
4
```

Note

In the first sample the performances number 2 and 3 are amazing.

In the second sample the performances number 2, 4, 9 and 10 are amazing.

B. Combination

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Ilya plays a card game by the following rules.

A player has several cards. Each card contains two non-negative integers inscribed, one at the top of the card and one at the bottom. At the beginning of the round the player chooses one of his cards to play it. If the top of the card contains number a_i, and the bottom contains number b_i, then when the player is playing the card, he gets a_i points and also gets the opportunity to play additional b_i cards. After the playing the card is discarded.

More formally: let's say that there is a counter of the cards that can be played. At the beginning of the round the counter equals one. When a card is played, the counter decreases by one for the played card and increases by the number b_i, which is written at the bottom of the card. Then the played card is discarded. If after that the counter is not equal to zero, the player gets the opportunity to play another card from the remaining cards. The round ends when the counter reaches zero or the player runs out of cards.

Of course, Ilya wants to get as many points as possible. Can you determine the maximum number of points he can score provided that you know his cards?

Input

The first line contains a single integer n $(1 \le n \le 1000)$ — the number of cards Ilya has.

Each of the next n lines contains two non-negative space-separated integers — a_i and b_i $(0 \le a_i, b_i \le 10^4)$ — the numbers, written at the top and the bottom of the i-th card correspondingly.

Output

Print the single number — the maximum number of points you can score in one round by the described rules.

Sample test(s)

input

```
2
1 0
2 0
```

output

```
2
```

input

```
3
1 0
2 0
0 2
```

output

```
3
```

Note

In the first sample none of two cards brings extra moves, so you should play the one that will bring more points.

In the second sample you should first play the third card that doesn't bring any points but lets you play both remaining cards.

C. Hometask

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Sergey attends lessons of the N-ish language. Each lesson he receives a hometask. This time the task is to translate some sentence to the N-ish language. Sentences of the N-ish language can be represented as strings consisting of lowercase Latin letters without spaces or punctuation marks.

Sergey totally forgot about the task until half an hour before the next lesson and hastily scribbled something down. But then he recollected that in the last lesson he learned the grammar of N-ish. The spelling rules state that N-ish contains some "forbidden" pairs of letters: such letters can never occur in a sentence next to each other. Also, the order of the letters doesn't matter (for example, if the pair of letters "`ab`" is forbidden, then any occurrences of substrings "`ab`" and "`ba`" are also forbidden). Also, each pair has different letters and each letter occurs in no more than one forbidden pair.

Now Sergey wants to correct his sentence so that it doesn't contain any "forbidden" pairs of letters that stand next to each other. However, he is running out of time, so he decided to simply cross out some letters from the sentence. What smallest number of letters will he have to cross out? When a letter is crossed out, it is "removed" so that the letters to its left and right (if they existed), become neighboring. For example, if we cross out the first letter from the string "`aba`", we get the string "`ba`", and if we cross out the second letter, we get "`aa`".

Input

The first line contains a non-empty string s, consisting of lowercase Latin letters — that's the initial sentence in N-ish, written by Sergey. The length of string s doesn't exceed 10^5.

The next line contains integer k $(0 \le k \le 13)$ — the number of forbidden pairs of letters.

Next k lines contain descriptions of forbidden pairs of letters. Each line contains exactly two different lowercase Latin letters without separators that represent the forbidden pairs. It is guaranteed that each letter is included in no more than one pair.

Output

Print the single number — the smallest number of letters that need to be removed to get a string without any forbidden pairs of neighboring letters. Please note that the answer always exists as it is always possible to remove all letters.

Sample test(s)

input

```
ababa
1
ab
```

output

```
2
```

input

```
codeforces
2
do
cs
```

output

```
1
```

Note

In the first sample you should remove two letters `b`.

In the second sample you should remove the second or the third letter. The second restriction doesn't influence the solution.

D. Colliders

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

By 2312 there were n Large Hadron Colliders in the inhabited part of the universe. Each of them corresponded to a single natural number from 1 to n. However, scientists did not know what activating several colliders simultaneously could cause, so the colliders were deactivated.

In 2312 there was a startling discovery: a collider's activity is safe if and only if all numbers of activated colliders are pairwise relatively prime to each other (two numbers are relatively prime if their greatest common divisor equals 1)! If two colliders with relatively nonprime numbers are activated, it will cause a global collapse.

Upon learning this, physicists rushed to turn the colliders on and off and carry out all sorts of experiments. To make sure than the scientists' quickness doesn't end with big trouble, the Large Hadron Colliders' Large Remote Control was created. You are commissioned to write the software for the remote (well, you do not expect anybody to operate it manually, do you?).

Initially, all colliders are deactivated. Your program receives multiple requests of the form "activate/deactivate the i-th collider". The program should handle requests in the order of receiving them. The program should print the processed results in the format described below.

To the request of "`+ i`" (that is, to activate the i-th collider), the program should print exactly one of the following responses:

- "`Success`" if the activation was successful.
- "`Already on`", if the i-th collider was already activated before the request.
- "`Conflict with j`", if there is a conflict with the j-th collider (that is, the j-th collider is on, and numbers i and j are not relatively prime). In this case, the i-th collider shouldn't be activated. If a conflict occurs with several colliders simultaneously, you should print the number of any of them.

The request of "`- i`" (that is, to deactivate the i-th collider), should receive one of the following responses from the program:

- "`Success`", if the deactivation was successful.
- "`Already off`", if the i-th collider was already deactivated before the request.

You don't need to print quotes in the output of the responses to the requests.

Input

The first line contains two space-separated integers n and m $(1 \le n, m \le 10^5)$ — the number of colliders and the number of requests, correspondingly.

Next m lines contain numbers of requests, one per line, in the form of either "`+ i`" (without the quotes) — activate the i-th collider, or "`- i`" (without the quotes) — deactivate the i-th collider $(1 \le i \le n)$.

Output

Print m lines — the results of executing requests in the above given format. The requests should be processed in the order, in which they are given in the input. Don't forget that the responses to the requests should be printed without quotes.

Sample test(s)

input

```
10 10
+ 6
+ 10
+ 5
- 10
- 5
- 6
+ 10
+ 3
+ 6
+ 3
```

output

```
Success
Conflict with 6
Success
Already off
Success
Success
Success
Success
Conflict with 10
Already on
```

Note

Note that in the sample the colliders don't turn on after the second and ninth requests. The ninth request could also receive response "`Conflict with 3`".

E. Double Profiles

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have been offered a job in a company developing a large social network. Your first task is connected with searching profiles that most probably belong to the same user.

The social network contains n registered profiles, numbered from 1 to n. Some pairs there are friends (the "friendship" relationship is mutual, that is, if i is friends with j, then j is also friends with i). Let's say that profiles i and j ($i \neq j$) are *doubles*, if for any profile k ($k \neq i$, $k \neq j$) one of the two statements is true: either k is friends with i and j, or k isn't friends with either of them. Also, i and j can be friends or not be friends.

Your task is to count the number of different unordered pairs (i, j), such that the profiles i and j are doubles. Note that the pairs are unordered, that is, pairs (a, b) and (b, a) are considered identical.

Input

The first line contains two space-separated integers n and m ($1 \le n \le 10^6$, $0 \le m \le 10^6$), — the number of profiles and the number of pairs of friends, correspondingly.

Next m lines contains descriptions of pairs of friends in the format "v u", where v and u ($1 \le v, u \le n$, $v \neq u$) are numbers of profiles that are friends with each other. It is guaranteed that each unordered pair of friends occurs no more than once and no profile is friends with itself.

Output

Print the single integer — the number of unordered pairs of profiles that are doubles.

Please do not use the `%lld` specificator to read or write 64-bit integers in C++. It is preferred to use the `%I64d` specificator.

Sample test(s)

input
```
3 3
1 2
2 3
1 3
```
output
```
3
```

input
```
3 0
```
output
```
3
```

input
```
4 1
1 3
```
output
```
2
```

Note

In the first and second sample any two profiles are doubles.

In the third sample the doubles are pairs of profiles $(1, 3)$ and $(2, 4)$.