A. 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.

B. 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`".

C. 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 $(2, 4)$.

D. Flatland Fencing

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

The King of Flatland will organize a knights' tournament! The winner will get half the kingdom and the favor of the princess of legendary beauty and wisdom. The final test of the applicants' courage and strength will be a fencing tournament. The tournament is held by the following rules: the participants fight one on one, the winner (or rather, the survivor) transfers to the next round.

Before the battle both participants stand at the specified points on the Ox axis with integer coordinates. Then they make moves in turn. The first participant moves first, naturally. During a move, the first participant can transfer from the point x to any integer point of the interval $[x + a; x + b]$. The second participant can transfer during a move to any integer point of the interval $[x - b; x - a]$. That is, the options for the players' moves are symmetric (note that the numbers a and b are not required to be positive, and if $a \le 0 \le b$, then staying in one place is a correct move). At any time the participants can be located arbitrarily relative to each other, that is, it is allowed to "jump" over the enemy in any direction. A participant wins if he uses his move to transfer to the point where his opponent is.

Of course, the princess has already chosen a husband and now she wants to make her sweetheart win the tournament. He has already reached the tournament finals and he is facing the last battle. The princess asks the tournament manager to arrange the tournament finalists in such a way that her sweetheart wins the tournament, considering that both players play optimally. However, the initial location of the participants has already been announced, and we can only pull some strings and determine which participant will be first and which one will be second. But how do we know which participant can secure the victory? Alas, the princess is not learned in the military affairs... Therefore, she asks you to determine how the battle will end considering that both opponents play optimally. Also, if the first player wins, your task is to determine his winning move.

Input

The first line contains four space-separated integers — x_1, x_2, a and b ($x_1 \ne x_2, a \le b, -10^9 \le x_1, x_2, a, b \le 10^9$) — coordinates of the points where the first and the second participant start, and the numbers that determine the players' moves, correspondingly.

Output

On the first line print the outcome of the battle as "`FIRST`" (without the quotes), if both players play optimally and the first player wins. Print "`SECOND`" (without the quotes) if the second player wins and print "`DRAW`" (without the quotes), if nobody is able to secure the victory.

If the first player wins, print on the next line the single integer x — the coordinate of the point where the first player should transfer to win. The indicated move should be valid, that is, it should meet the following condition: $x_1 + a \le x \le x_1 + b$. If there are several winning moves, print any of them. If the first participant can't secure the victory, then you do not have to print anything.

Sample test(s)

input
```
0 2 0 4
```
output
```
FIRST
2
```

input
```
0 2 1 1
```
output
```
SECOND
```

input
```
0 2 0 1
```
output
```
DRAW
```

Note

In the first sample the first player can win in one move.

In the second sample the first participant must go to point 1, where the second participant immediately goes and wins.

In the third sample changing the position isn't profitable to either participant, so nobody wins.

E. Martian Colony

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The first ship with the Earth settlers landed on Mars. The colonists managed to build n necessary structures on the surface of the planet (which can be regarded as a plane, and the construction can be regarded as points on it). But one day the scanners recorded suspicious activity on the outskirts of the colony. It was decided to use the protective force field generating system to protect the colony against possible trouble.

The system works as follows: the surface contains a number of generators of the field (they can also be considered as points). The active range of each generator is a circle of radius r centered at the location of the generator (the boundary of the circle is also included in the range). After the system is activated, it stretches the protective force field **only over the part of the surface, which is within the area of all generators' activity**. That is, the protected part is the **intersection** of the generators' active ranges.

The number of generators available to the colonists is not limited, but the system of field generation consumes a lot of energy. More precisely, the energy consumption does not depend on the number of generators, but it is directly proportional to the **area**, which is protected by the field. Also, it is necessary that all the existing buildings are located within the protected area.

Determine the smallest possible area of the protected part of the surface containing all the buildings.

Input

The first line contains two integers n and r ($1 \le n \le 10^5$, $1 \le r \le 50000$) — the number of buildings and the active ranges of the generators, correspondingly.

Next n lines contains the buildings' coordinates. The $i + 1$-th ($1 \le i \le n$) line contains two real numbers with at most three digits after the decimal point x_i and y_i ($|x_i|, |y_i| \le 50000$) — coordinates of the i-th building. It is guaranteed that no two buildings are located at the same point, and no two different buildings are located closer than 1.

It is guaranteed that there exists a circle with radius r that contains all the buildings.

Output

Print the single real number — the minimum area of the protected part containing all the buildings. The answer is accepted if absolute or relative error doesn't exceed 10^{-4}.

Sample test(s)

input

```
3 5
0.00 0.000
0.0 8.00
6 8.00
```

output

```
78.5398163397
```

input

```
4 1000
0.0 0.0
0 2.00
2.00 2
2.0 0.00
```

output

```
4.0026666140
```

input

```
4 5
3.00 0.0
-3 0.00
0.000 1
0.0 -1.00
```

output

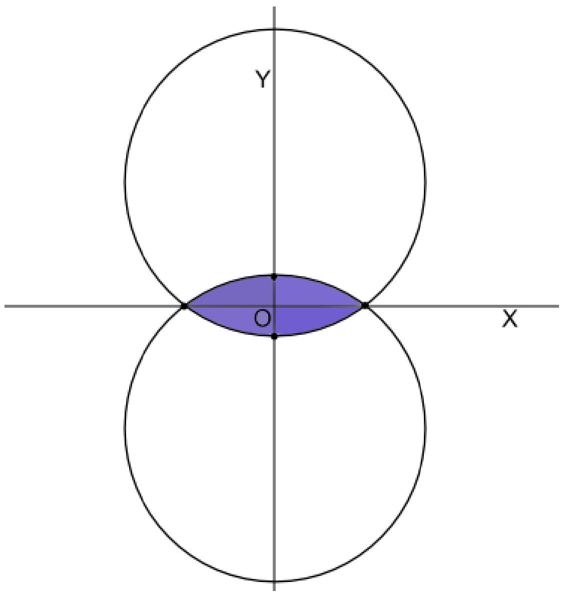
```
8.1750554397
```

Note

In the first sample the given radius equals the radius of the circle circumscribed around the given points. That's why the circle that corresponds to it is the sought area. The answer is 25π.

In the second sample the area nearly coincides with the square which has vertexes in the given points.

The area for the third sample is shown on the picture below.


Y
O
X