Pacific Northwest Region Programming Contest Division 1

November 11th, 2017

Reminders

- For all problems, read the input data from standard input and write the results to standard output.
- In general, when there is more than one integer or word on an input line, they will be separated from each other by exactly one space. No input lines will have leading or trailing spaces, and tabs will never appear in any input.
- Platform is as follows:

```
Ubuntu 16.04.3 LTS Linux (64-bit)
GNOME

vi/vim
gvim
emacs
gedit
geany
kate

Java OpenJDK version 1.8.0_131
C gcc 5.4.0
C++ g++ 5.4.0
Python 2.7.10 (implemented using PyPy 5.1.2).
CPython 3.5.2.
C# via Mono 4.2.1.

Eclipse 4.7 (Oxygen), configured with:
   Java
   C/C++
PyDev
IntelliJ (IDEA Community Edition 2017.2.3), configured with:
   Java (version TBD)
Lion (version 2017.2.2), configured with
   C/C++ (version TBD)
Pycharm Community Edition Python IDE version 2017.2.2
Code::Blocks (version 13.12+dfsg-4), configured with
   Java (OpenJDK version 1.8.0_131)
   C/C++ (CDT 9.3.0 with Ubuntu 5.4.0-6ubuntu116.04.4 5.4.0 20160609)
MonoDevelop 5.10.0.871-2
```

- Compiler options are as follows:

```
gcc -g -O2 -std=gnu11 -static $* -lm
g++ -g -O2 -std=gnu++14 -static $*
javac -encoding UTF-8 -sourcepath . -d . $*
```

```
python2 -m py_compile $*
python3 -m py_compile $*
mcs $*
```

- Execution options are as follows:

```
java -XX:+UseSerialGC -Xss64m -Xms1920m -Xmx1920m $*
pypy $*
mono $*
```

- Python may not have sufficient performance for many of the problems; use it at your discretion. This is especially true of Python 3.

Odd Palindrome

We say that a string is *odd* if and only if all palindromic substrings of the string have odd length.

Given a string s, determine if it is *odd* or not.

A substring of a string s is a nonempty sequence of consecutive characters from s. A palindromic substring is a substring that reads the same forwards and backwards.

Input

The input consists of a single line containing the string s ($1 \leq |s| \leq 100$).

It is guaranteed that s consists of lowercase ASCII letters (‘`a`’–‘`z`’) only.

Output

If s is *odd*, then print “`Odd.`” on a single line (without quotation marks). Otherwise, print “`Or not.`” on a single line (without quotation marks).

Sample Input and Output

`amanaplanacanalpanama`	`Odd.`

`madamimadam`	`Odd.`

`annamyfriend`	`Or not.`

`nopalindromes`	`Odd.`

Problem B — limit 4 seconds

Enlarging Enthusiasm

You are watching a singing contest featuring n singers, and the final round has just finished. Currently, the ith singer has p_i points. Each singer currently has a distinct number of points.

It is time to determine the final rankings. The judges are about to award additional points to each of the singers. The total points given in the final round across all singers will sum to x. More specifically, for each singer, the judges will choose numbers $q_i \geq 1$ to assign each of the singers, so that $q_1 + \cdots + q_n = x$.

The judges would like to make the announcement of final results as exciting as possible. To do this, they will announce the scores q_i one by one, in increasing order. If they award the same number of points (q_i) to more than one singer, they will announce those scores in order of increasing p_i.

After each score announcement, the rankings will get updated. The announcement is called *exciting*, if after each score update, the singer with the highest number of points changes. The judges want to avoid any sort of controversy, so they will assign the points so that at any point during the score announcements, there is a unique singer with the highest number of points.

Help the judges determine the number of distinct final rankings that can be attained by assigning points in the final round such every score announcement is exciting.

Input

The first line of input contains two space-separated integers n and x ($1 \leq n \leq 12$; $1 \leq x \leq 700$).

The next line contains n integers p_i ($1 \leq p_i \leq 700$) separated by spaces.

It is guaranteed that all p_i are distinct.

Output

Print the number of different final rankings that can be achieved by assigning points such that every announcement is *exciting.*

For the first sample, there are three contestants A, B, and C. A currently has 3 points, B currently has 1 point, and C currently has 4 points. The judges can assign 12 points.

Suppose the judges chose to assign (q) 2 points to A, 7 points to B, and 3 points to C. The scores will be announced in increasing order of q. The following sequence of actions happen:

- A's score is announced. Her score becomes 5, and she is the new leader.
- C's score is announced. His score becomes 7, and he is the new leader.
- B's score is announced. His score becomes 8, and he is the new leader.

We can see this is an exciting set of announcements.

On the other hand, the judges choosing $q = [3, 3, 6]$ is not exciting, since the leader isn't unique after the first announcement. (We announce B first since they have the minimum p_i between the two contestants both getting 3 points from the judges.) Also, the $q = [6, 5, 1]$ is not exciting, since the leader doesn't change after the first announcement.

Given this scenario, three distinct final rankings that are possible (from highest ranked to lowest ranked):

- C, B, A, with judges $q = [2, 5, 5]$
- B, C, A, with judges $q = [2, 8, 2]$
- C, A, B, with judges $q = [4, 4, 4]$

Note that the order B, C, A is only counted once even though it can be realized in two different ways ($q = [2, 8, 2]$ and $q = [2, 7, 3]$).

Sample Input and Output

```
3 12
3 1 4
```

```
3
```

```
12 700
1 2 3 4 5 6 7 8 9 10 11 12
```

```
439084800
```

```
12 1
1 2 3 4 5 6 7 8 9 10 11 12
```

```
0
```

```
1 700
123
```

```
0
```

Problem C — limit 1 second

Fear Factoring

FEAR FACTORING

The Slivians are afraid of factoring; it's just, well, difficult.

Really, they don't even care about the factors themselves, just how much they sum to.

We can define $F(n)$ as the sum of all of the factors of n; so $F(6) = 12$ and $F(12) = 28$. Your task is, given two integers a and b with $a \leq b$, to calculate

$$S = \sum_{a \leq n \leq b} F(n).$$

Input

The input consists of a single line containing space-separated integers a and b ($1 \leq a \leq b \leq 10^{12}$; $b - a \leq 10^6$).

Output

Print S on a single line.

Sample Input and Output

`101 101`	`102`

`28 28`	`56`

`1 10`	`87`

`987654456799 987654456799`	`987654456800`

963761198400 963761198400	5531765944320

5260013877 5260489265	4113430571304040

PROBLEM D — LIMIT 2 SECONDS

Rainbow Roads

You are given a tree with n nodes (conveniently numbered from 1 to n). Each edge in this tree has one of n colors. A path in this tree is called a *rainbow* if all adjacent edges in the path have different colors. Also, a node is called *good* if every simple path with that node as one of its endpoints is a *rainbow* path.

Find all the *good* nodes in the given tree.

A simple path is a path that does not repeat any vertex or edge.

Input

The first line of input contains a single integer n ($1 \leq n \leq 50{,}000$).

Each of the next $n - 1$ lines contains three space-separated integers a_i, b_i, and c_i ($1 \leq a_i, b_i, c_i \leq n$; $a_i \neq b_i$), describing an edge of color c_i that connects nodes a_i and b_i.

It is guaranteed that the given edges form a tree.

Output

On the first line of the output, print k, the number of good nodes.

In the next k lines, print the indices of all good nodes in numerical order, one per line.

For the first sample, node 3 is good since all paths that have node 3 as an endpoint are rainbow. In particular, even though the path 3—4—5—6 has two edges of the same color (i.e. 3—4, 5—6), it is still rainbow since these edges are not adjacent.

Sample Input and Output

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

```
4
3
4
5
6
```

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

```
0
```

```
9
1 2 2
1 3 1
1 4 5
1 5 5
2 6 3
3 7 3
4 8 1
5 9 2
```

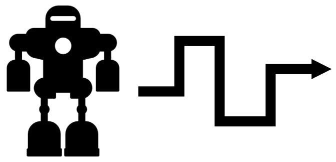
```
5
1
2
3
6
7
```

```
10
9 2 1
9 3 1
9 4 2
9 5 2
9 1 3
9 6 4
1 8 5
1 10 5
6 7 9
```

```
4
1
6
7
9
```

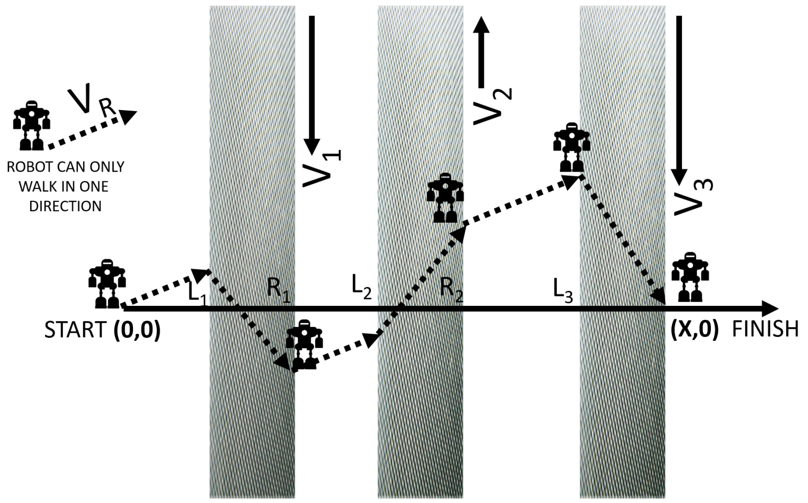
Problem E — limit 1 second

Straight Shot



You have a toy robot that walks straight at a constant speed v, and you wish for it to travel on the two-dimensional plane from $(0, 0)$ to $(X, 0)$. If the plane were empty, you could start the robot facing straight east from the origin, and it would walk there in X/v time. Unfortunately, between the start and the destination are n moving sidewalks, each moving directly north or south, which affect the robot's position while it is walking.

The direction that robot is facing is not changed by the sidewalks; the robot will face in the same orientation for the entire duration of its walk. These sidewalks are aligned with the y-axis and are infinitely long. You still must get the robot to go from start to finish, but you'll need to adjust the orientation of the robot at the start. Given that you choose this direction correctly, so that the robot arrives exactly at the destination, how long will it take the robot to get there?



One final caveat: You don't want the toy robot to walk for too long. If the robot cannot reach the destination in at most twice the time it would take in the absence of all moving sidewalks (*i.e.*, $2X/v$), indicate this.

Input

The first line consists of three space-separated numbers n, X, and v ($0 \leq n \leq 100$; $1 \leq X \leq$ 1,000,000; $1.0 \leq v \leq 100.0$). Note that v is not necessarily an integer.

Each of the next n lines contains three space-separated numbers l_i, r_i, and v_i ($0 \le l_1 < r_1 \le l_2 < r_2 \le \cdots \le l_n < r_n \le X$; $-100.0 \le v_i \le 100.0$), describing the ith moving sidewalk. The integer l_i denotes the left edge of the sidewalk, the integer r_i denotes the right edge of the sidewalk, and the decimal number v_i denotes the speed of the sidewalk. A positive speed means the sidewalk moves north, while a negative speed means the sidewalk moves south.

Output

If the robot cannot reach the destination in at most twice the time it would take in the absence of all moving sidewalks, output "`Too hard`" on a single line (without quotation marks).

Otherwise, output, on a single line, the travel time of the robot from the start to the destination, rounded and displayed to exactly three decimal places.

Sample Input and Output

```
1 806873 66
91411 631975 -57.5
```

```
15055.988
```

```
2 422193 100
38180 307590 86.4
366035 403677 -4.7
```

```
5043.896
```

```
1 670764 22.4
113447 642610 -64.8
```

```
Too hard
```

Problem F — limit 3 seconds

Distinct Distances

You're setting up a scavenger hunt that takes place in the two-dimensional plane.

You've already decided on n distinct points of interest labeled $p_1, \ldots, p_n$. The point p_i is located at integer coordinates (x_i, y_i).

You now want to choose a point q for the final location. This point must have finite coordinates, but it does not necessarily need to have integer coordinates. This point also can coincide with one of the original points p_i.

In order to make this final location interesting, you would like to minimize the number of unique distances from q to the other points.

More precisely, you would like to choose q that minimizes $|S(q)|$, where $S(q)$ is defined as the set

$$\{|q - p_1|, |q - p_2|, \ldots, |q - p_n|\}.$$

Here, the notation $|S(q)|$ means the number of elements in the set $S(q)$ and $|q - p_i|$ denotes the Euclidean distance between q and p_i. Note that $S(q)$ is a set, so if two or more distances $|q - p_i|$ are equal, they are counted as a single element in $S(q)$.

Given the coordinates of the points, find the minimum value of $|S(q)|$.

Warning: Use of inexact arithmetic may make it difficult to identify distances that are exactly equal.

Input

The first line of input contains a single integer n ($1 \leq n \leq 40$).

Each of the next n lines contains two space-separated integers x_i and y_i ($|x_i|, |y_i| \leq 300$), representing the coordinates of p_i.

Output

Output, on a single line, the minimum number of unique distances from q to all other points p_i.

For the first sample, we can let our point q be $(0, 0)$. All other points are distance 5 away. For the second sample, we can let q be $(1.5, 1.5)$.

Sample Input and Output

```
8
3 4
0 5
0 -5
5 0
-5 0
4 -3
3 -4
-4 3
```

```
1
```

```
4
0 0
1 1
2 2
3 3
```

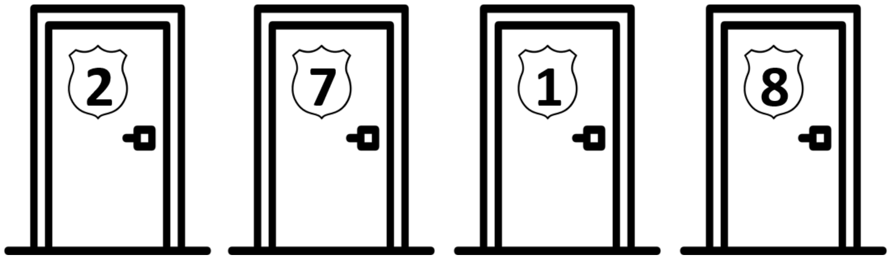
```
2
```

```
6
0 -5
1 0
-1 0
2 3
3 2
-3 0
```

```
3
```

Problem G — limit 3 seconds

Security Badge



You are in charge of the security for a large building, with n rooms and m doors between the rooms. The rooms and doors are conveniently numbered from 1 to n, and from 1 to m, respectively.

Door i opens from room a_i to room b_i, but not the other way around. Additionally, each door has a security code that can be represented as a range of numbers $[c_i, d_i]$.

There are k employees working in the building, each carrying a security badge with a unique, integer-valued badge ID between 1 and k. An employee is cleared to go through door i only when the badge ID x satisfies $c_i \leq x \leq d_i$.

Your boss wants a quick check of the security of the building. Given s and t, how many employees can go from room s to room t?

Input

The first line of input contains three space-separated integers n, m, and k ($2 \leq n \leq 1{,}000$; $1 \leq m \leq 5{,}000$; $1 \leq k \leq 10^9$).

The second line of input contains two space-separated integers s and t ($1 \leq s, t \leq n$; $s \neq t$).

Each of the next m lines contains four space-separated integers a_i, b_i, c_i, and d_i ($1 \leq a_i, b_i \leq n$; $1 \leq c_i \leq d_i \leq k; a_i \neq b_i$), describing door i.

For any given pair of rooms a, b there will be at most one door from a to b (but there may be both a door from a to b and a door from b to a).

Output

Print, on a single line, the number of employees who can reach room t starting from room s.

Sample Input and Output

```
4 5 10
3 2
1 2 4 7
3 1 1 6
3 4 7 10
2 4 3 5
4 2 8 9
```

```
5
```

```
4 5 9
1 4
1 2 3 5
1 3 6 7
1 4 2 3
2 4 4 6
3 4 7 9
```

```
5
```

Problem H — limit 3 seconds

Avoiding Airports

David is looking to book some travel all over the world. There are n countries that he can visit, and m flights that are available. The ith flight goes from country a_i to country b_i. It departs at time s_i, and lands at time e_i.

David is currently in the airport in country 1, and the current time is 0, and he would like to travel to country n. He does not care about the total amount of time needed to travel, but he really hates waiting in the airport. If he waits t time units in an airport, he gains t^2 units of frustration. Help him find an itinerary that minimizes the total frustration.

Input

The first line of input contains two space-separated integers n and m ($2 \le n \le 200{,}000; 1 \le m \le 200{,}000$).

Each of the next m lines contains four space-separated integers a_i, b_i, s_i, and e_i ($1 \le a_i, b_i \le n$; $0 \le s_i \le e_i \le 10^6$). This describes a flight from country a_i to country b_i that departs at time s_i and arrives at time e_i.

A flight might have the same departure and arrival country.

No two flights will have the same arrival time, or have the same departure time. In addition, no flight will have the same arrival time as the departure time of another flight. Finally, it is guaranteed that there will always be a way for David to arrive at his destination.

Output

Print, on a single line, the minimum total frustration.

In the first sample, it is optimal to take this sequence of flights:

- Flight 5. Goes from airport 1 to airport 2, departing at time 3, arriving at time 8.
- Flight 3. Goes from airport 2 to airport 1, departing at time 9, arriving at time 12.
- Flight 7. Goes from airport 1 to airport 3, departing at time 13, arriving at time 27.
- Flight 8. Goes from airport 3 to airport 5, deparing at time 28, arriving at time 100.

The frustration for each wait is $3^2, 1^2, 1^2$, and 1^2, respectively. Thus, the total frustration is 12.

Note that there is an itinerary that gets David to his destination faster. However, that itinerary has a higher total frustration.

Sample Input and Output

```
5 8
1 2 1 10
2 4 11 16
2 1 9 12
3 5 28 100
1 2 3 8
4 3 20 21
1 3 13 27
3 5 23 24
```

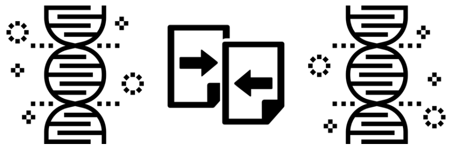
```
12
```

```
3 5
1 1 10 20
1 2 30 40
1 2 50 60
1 2 70 80
2 3 90 95
```

```
1900
```

Problem I — limit 1 second

Long Long Strings



To store long DNA sequences, your company has developed a `LongLongString` class that can store strings with more than ten billion characters. The class supports two basic operations:

- `Ins(`p`, `c`)`: Insert the character c at position p.
- `Del(`p`)`: Delete the character at position p.

A DNA editing program is written as a series of `Ins` and `Del` operations. Your job is to write a program that compare two DNA editing programs and determine if they are identical, *i.e.*, when applied to any sufficiently long string, whether the end result is the same. For example:

- `Del(1) Del(2)` and `Del(3) Del(1)` are identical.
- `Del(2) Del(1)` and `Del(1) Del(2)` are different.
- An empty sequence and `Ins(1, `x`) Del(1)` are identical.
- `Ins(14, `b`) Ins(14, `a`)` and `Ins(14, `a`) Ins(15, `b`)` are identical.
- `Ins(14, `a`) Ins(15, `b`)` and `Ins(14, `b`) Ins(15, `a`)` are different.

Input

Input will consist of the descriptions of two DNA editing programs. Each program will consist of some number of operations (between 0 and 2,000). Each operation will be given on its own line. The first character of the line will be `D` for a `Del` operation, `I` for an `Ins` operation, or `E` marking the end of the program.

A `Del` operation will have the `D` character, followed by a space, and then a single integer between 1 and 10^{10}, indicating the character position to delete. All characters after this deleted character will be shifted one position lower.

An `Ins` operation will have the `I` character, followed by a space, and then a single integer between 1 and 10^{10}, indicating the location to insert the new character; all pre-existing characters with this index and higher will be shifted one position higher. Following this integer will be another space and then an uppercase alphabetic character that is the character to insert.

Output

If the two programs are identical, print “`0`” on a single line (without quotation marks). Otherwise, print “`1`” on a single line (without quotation marks).

Sample Input and Output

```
D 1
D 2
E
D 3
D 1
E
```

```
0
```

```
D 2
D 1
E
D 1
D 2
E
```

```
1
```

```
I 1 X
D 1
E
E
```

```
0
```

```
I 14 B
I 14 A
E
I 14 A
I 15 B
E
```

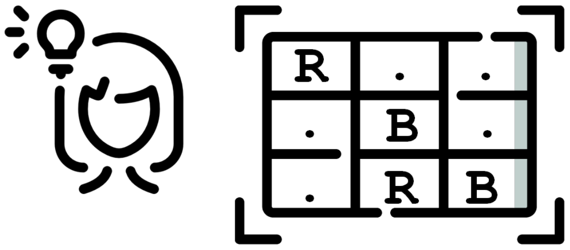
```
0
```

```
I 14 A
I 15 B
E
I 14 B
I 15 A
E
```

```
1
```

Problem J — limit 1 second

Grid Coloring



You have an m-by-n grid of squares that you wish to color. You may color each square either red or blue, subject to the following constraints:

- Every square must be colored.
- Colors of some squares are already decided (red or blue), and cannot be changed.
- For each blue square, all squares in the rectangle from the top left of the grid to that square must also be blue.

Given these constraints, how many distinct colorings of the grid are there? The grid cannot be rotated.

Input

The first line of input consists of two space-separated integers m and n ($1 \leq m, n \leq 30$).

Each of the next m lines contains n characters, representing the grid. Character '`B`' indicates squares that are already colored blue. Similarly, '`R`' indicates red squares. Character '`.`' indicates squares that are not colored yet.

Output

Print, on a single line, the number of distinct colorings possible.

For the first sample, the 6 ways are:

```
BB      BB      BR      BR      BB      BB
BB      BR      BR      BR      BR      BB
BR      BR      BR      RR      RR      RR
```

Sample Input and Output

```
3 2
..
B.
.R
```

```
6
```

```
7 6
......
.....B
.B..R.
......
...B..
.R....
...R..
```

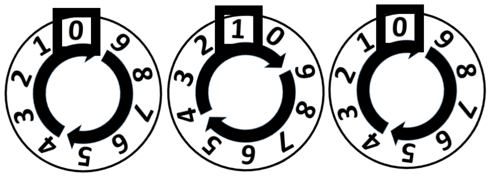
```
3
```

```
2 2
R.
.B
```

```
0
```

Problem K — limit 1 second

Spinning Up Palindromes



"Sabotage!", exclaimed J. R. Diddly, president and founder of Diddly Widgets Inc.

"Vandalism, perhaps. Nothing's actually been damaged." responded Robert Lackey, the chief accountant.

Both were staring up at the large counter suspended above the factory floor, a counter that had faithfully recorded the number of widgets that had come off the assembly line since the factory was opened. But someone had changed the number being displayed so that it formed...

"It's a palindrome." said Lackey. "It reads the same forwards as backwards."

"What I don't understand," said Diddly, "is why our security guards didn't catch the vandals during their regular sweeps. It must have taken them hours to click forward to this new number, one step at a time."

"No." replied Lackey. "Although we only advance the rightmost digit each time a new widget is built, it's possible to spin any of the digits. With a little planning, this might have taken only a few seconds."

Consider a digital counter consisting of k wheels, each showing a digit from `0` to `9`. Each wheel is mounted so that it can advance to the next digit in a single step, *e.g.*, from `3` to `4`, or from `8` to `9`.

It is also possible to advance from digit `9` to digit `0`. However, when this happens, the wheel on its immediate left will also advance to the next digit automatically. This can have a cascade effect on multiple wheels to the left, but they all happen in a single step.

Given the current setting of the counter, find the smallest number of steps until one can reach a palindrome. The palindrome must respect leading zeros, *e.g.*, `0011` is not a palindrome.

For example, for input `610`, it takes four steps. This can be done by incrementing the `6` wheel four times, resulting in `010`.

Input

The first line of input contains a string of k digits ($1 \leq k \leq 40$), representing the current setting of the counter.

Note that the input may contain leading zeros.

Output

Print, on a single line, the minimum number of wheel advances necessary to produce a palindrome.

Sample Input and Output

0	0
009990001	3
29998	5
610	4
981	2
9084194700940903797191718247801197019268	54

Problem L — limit 1 second

Delayed Work

You own a company that hires painters to paint houses. You can hire as many painters as you want, but for every painter you hire, you have to pay X dollars (independent of how long the painter works on the house). In addition, you have to pay a penalty of $D \cdot P$ dollars overall if it takes D days to finish painting the house. This penalty does not depend on the number of painters you hire; furthermore, even if the time taken is not a whole number of days, you are only penalized for the exact time taken.

All painters paint at the same rate. One painter can finish painting the house in K days. The painters cooperate so well that it will only take K/M days for M painters to finish painting the house.

What is the minimum amount of money you will have to pay, including what you pay to the painters and the cost penalty, to get a house painted?

Input

The input consists of a single line containing three space-separated integers K, P, and X ($1 \leq K, P, X \leq 10{,}000$).

Output

Print, on a single line, the minimum cost to have the house painted, rounded and displayed to exactly three decimal places.

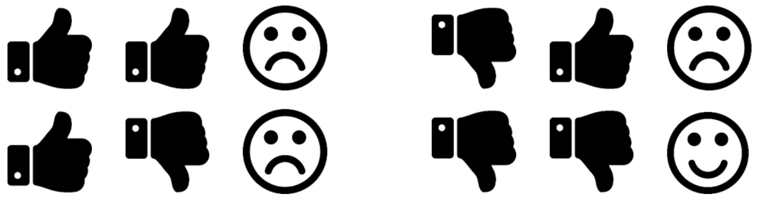
Sample Input and Output

`31 41 59`	`549.200`

`3 4 5`	`16.000`

Problem M — limit 2 seconds

Unsatisfying



Usually a computer scientist tries to satisfy some constraints. This time you will try to make some logical statements unsatisfiable.

You will be given a list of logical statements of the following form:

$$\begin{aligned} p_1 &\vee p_2, \\ \neg p_2 &\vee p_3, \\ p_3 &\vee \neg p_4. \end{aligned}$$

Each of the p_i is a proposition that can be either `true` or `false`. The disjunction operator $\vee$ can be interpreted as logical OR. The $\neg$ symbol is negation, which negates the value of the subsequent proposition from `true` to `false`, and vice versa.

To satisfy a list of logical statements, you must assign either `true` or `false` to each of the propositions such that every disjunction in the list evaluates to `true`.

Your task is to add disjunctions to the list to make the list of statements unsatisfiable. But the additional disjunctions you add cannot use the negation symbol!

All disjunctions (both those given and ones you add) must have exactly 2 terms.

Input

The first line of input contains two space-separated integers n and m ($1 \leq n, m \leq 2{,}000$), with n representing the number of propositions and m representing the number of disjunctions.

Each of the next m lines contains two space-separated integers a_i and b_i ($1 \leq |a_i|, |b_i| \leq n$), which describes the two propositions of the ith disjunction in the given list. A positive value a_i, for example, represents the proposition p_{a_i}. On the other hand, if a_i is negative, it represents the negated proposition, namely $\neg p_{|a_i|}$.

The second sample input corresponds to the following list of logical statements:

$$\begin{aligned} p_1 &\vee p_2, \\ \neg p_1 &\vee \neg p_3, \\ \neg p_2 &\vee p_3, \\ p_3 &\vee \neg p_4, \\ \neg p_2 &\vee \neg p_3. \end{aligned}$$

Output

Output, on a single line, a single integer representing the minimum number of additional disjunctions necessary to make the list unsatisfiable. If it is not possible to make the list unsatisfiable, print `-1` instead. Each disjunction you add must have two (not necessarily distinct) propositions, and you may not use negated propositions.

In the second sample case, adding a disjunction $p_2 \vee p_2$ makes the list unsatisfiable.

Sample Input and Output

```
2 1
1 2
```

```
-1
```

```
4 5
1 2
-1 -3
-2 3
3 -4
-2 -3
```

```
1
```