Pacific Northwest Region Programming Contest Division 2

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.

Problem N — limit 1 second

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

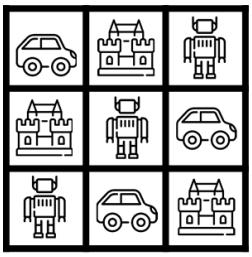
If s is *odd*, then print "`Odd.`" on a single line (without quotation marks). Otherwise, print "`Or not.`" on a single line (without quotation marks).

Sample Input and Output

`amanaplanacanalpanama`	`Odd.`
`madamimadam`	`Odd.`
`annamyfriend`	`Or not.`
`nopalindromes`	`Odd.`

Problem O — limit 1 second

Latin Squares



A Latin Square is an n-by-n array filled with n different digits, each digit occurring exactly once in each row and once in each column. (The name "Latin Square" was inspired by the work of Leonhard Euler, who used Latin characters in his papers on the topic.)

A Latin Square is said to be in *reduced form* if both its top row and leftmost column are in their natural order. The natural order of a set of digits is by increasing value.

Your team is to write a program that will read an n-by-n array, and determine whether it is a Latin Square, and if so, whether it is in reduced form.

Input

The first line of input contains a single integer n ($2 \leq n \leq 36$). Each of the next n lines contains n digits in base n, with the normal digits '`0`' through '`9`' for digit values below 10 and uppercase letters '`A`' through '`Z`' representing digit values 10 through 35. All digits will be legal for base n; for instance, if n is 3, the only legal characters in the n input lines describing the square will be '`0`', '`1`', and '`2`'.

Output

If the given array is not a Latin Square, print "`No`" on a single line (without quotation marks). If it is a Latin Square, but not in reduced form, print "`Not Reduced`" on a single line (without quotation marks). If it is a Latin Square in reduced form, print "`Reduced`" on a single line (without quotation marks).

Sample Input and Output

3 012 120 201	Reduced

4 3210 0123 2301 1032	Not Reduced

11 0123458372A A9287346283 0285475A834 84738299A02 1947584037A 65848430002 038955873A8 947530200A8 93484721084 95539A92828 04553883568	No

Problem P — limit 1 second

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

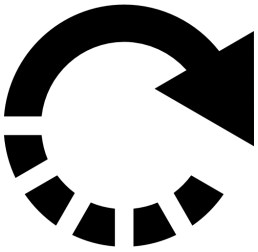
101 101	102
28 28	56
1 10	87
987654456799 987654456799	987654456800

963761198400 963761198400	5531765944320

5260013877 5260489265	4113430571304040

Problem Q — limit 1 second

Halfway



A friend of yours has written a program that compares every pair of a list of items. With n items, it works as follows. First, it prints a 1, and it compares item 1 to items $2, 3, 4, \ldots, n$. It then prints a 2, and compares item 2 to items $3, 4, 5, \ldots, n$. It continues like that until every pair of items has been compared exactly once. If it compares item x to item y, it will not later compare item y to item x. Also, it does not compare any item to itself.

Your friend wants to know when his program is *halfway done*. For a program that makes an odd number of total comparisons, this is when it is doing the middle comparison. For a program that makes an even number of total comparisons, this is when it is doing the first of the two middle comparisons.

What will the last number printed be when the program is halfway done?

Note that since the earlier items have more comparisons than the later items, the answer is not simply $n/2$.

Input

The input consists of a single line containing the integer n ($2 \leq n \leq 10^9$).

Output

Print, on a single line, the last number your friend's program prints when it is halfway done.

Sample Input and Output

4	1

7	2

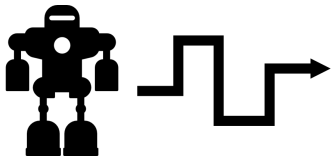
10	3

1919	562

290976843	85225144

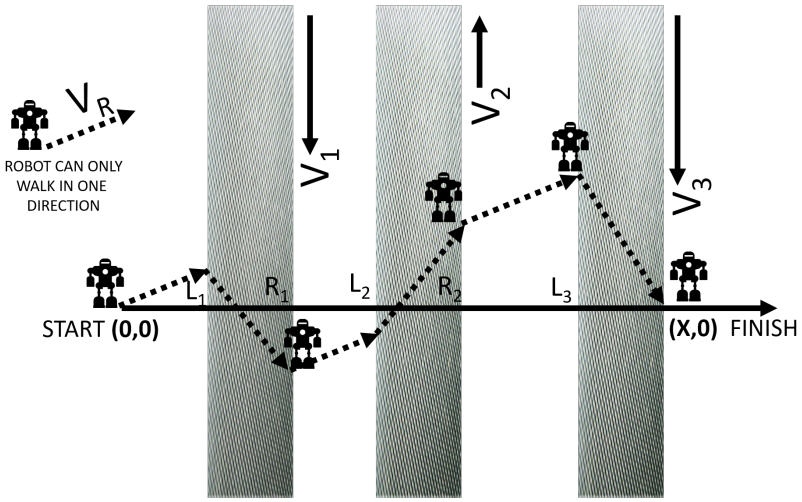
Problem R — limit 1 second

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 ($0 \leq l_1 < r_1 \leq l_2 < r_2 \leq \cdots \leq l_n < r_n \leq X$; $-100.0 \leq v_i \leq 100.0$), 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 S — limit 1 second

Purple Rain

Purple rain falls in the magic kingdom of Linearland which is a straight, thin peninsula.

On close observation however, Professor Nelson Rogers finds that the purple rain is actually a mix of red and blue raindrops.

In his zeal, he records the location and color of the raindrops in different locations along the peninsula. Looking at the data, Professor Rogers wants to know which part of Linearland had the "least" purple rain.

After some thought, he decides to model this problem as follows. Divide the peninsula into n sections and number them west to east from 1 to n. Then, describe the raindrops as a sequence of `R` and `B`, depending on whether the rainfall in each section is primarily red or blue. Finally, find a subsequence of contiguous sections where the difference between the number of `R` and the number of `B` is maximized.

Input

The input consists of a single line containing a string of n characters ($1 \leq n \leq 10^5$), describing the color of the raindrops in sections 1 to n.

It is guaranteed that the string consists of uppercase ASCII letters '`R`' and '`B`' only.

Output

Print, on a single line, two space-separated integers that describe the starting and ending positions of the part of Linearland that had the least purple rain. These two numbers should describe an inclusive range; both numbers you print describe sections included in the range.

If there are multiple possible answers, print the one that has the westernmost starting section. If there are multiple answers with the same westernmost starting section, print the one with the westernmost ending section.

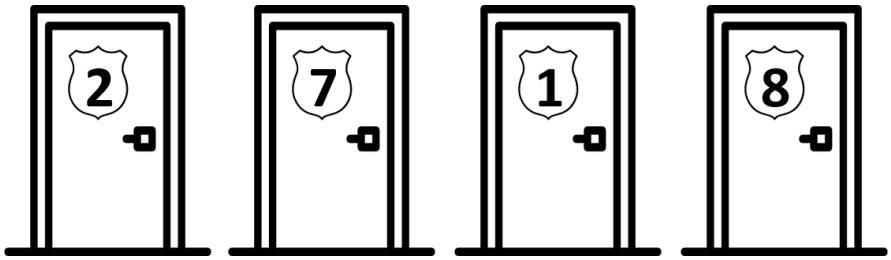
Sample Input and Output

BBRRBRRBRB	3 7

BBRBBRRB	1 5

Problem T — 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 U — limit 1 second

Unloaded Die

Consider a six-sided die, with sides labeled 1 through 6. We say the die is *fair* if each of its sides is equally likely to be face up after a roll. We say the die is *loaded* if it isn't fair. For example, if the side marked 6 is twice as likely to come up as than any other side, we are dealing with a loaded die.

For any die, define the *expected result* of rolling the die to be equal to the average of the values of the sides, weighted by the probability of those sides coming up. For example, all six sides of a fair die are equally likely to come up, and thus the expected result of rolling it is $(1+2+3+4+5+6)/6 = 3.5$.

You are given a loaded die, and you would like to *unload* it to make it more closely resemble a fair die. To do so, you can erase the number on one of the sides, and replace it with a new number which does not need to be an integer or even positive. You want to do so in such a way that

- The expected result of rolling the die is 3.5, just like a fair die.
- The difference between the old label and the new label on the side you change is as small as possible.

Input

The input consists of a single line containing six space-separated nonnegative real numbers $v_1 \ldots v_6$, where v_i represents the probability that side i (currently labeled by the number i) is rolled.

It is guaranteed that the given numbers will sum to 1.

Output

Print, on a single line, the absolute value of the difference between the new label and old label, rounded and displayed to exactly three decimal places.

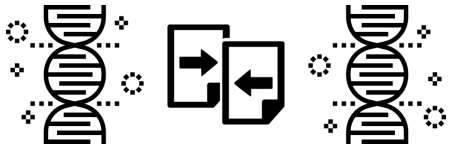
Sample Input and Output

0.16666 0.16667 0.16667 0.16666 0.16667 0.16667	0.000

0.2 0.2 0.1 0.2 0.2 0.1	1.000

Problem V — 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

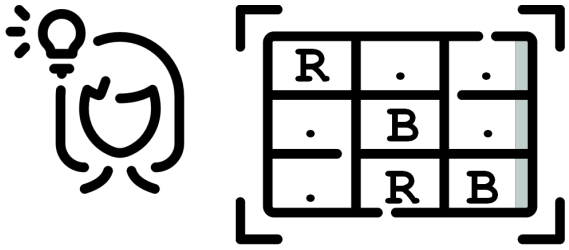
If the two programs are identical, print "`0`" on a single line (without quotation marks). Otherwise, print "`1`" on a single line (without quotation marks).

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 W — 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 X — limit 1 second

Star Arrangements

The recent vote in Puerto Rico favoring United States statehood has made flag makers very excited. An updated flag with 51 stars rather than the current one with 50 would cause a huge jump in U.S. flag sales. The current pattern for 50 stars is five rows of 6 stars, interlaced with four offset rows of 5 stars. The rows alternate until all stars are represented.

```
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
```

This pattern has the property that adjacent rows differ by no more than one star. We represent this star arrangement compactly by the number of stars in the first two rows: `6,5`.

A 51-star flag that has the same property can have three rows of 9 stars, interlaced with three rows of 8 stars (with a compact representation of `9,8`). Conversely, if a state were to leave the union, one appealing representation would be seven rows of seven stars (`7,7`).

A flag pattern is *visually appealing* if it satisfies the following conditions:

- Every other row has the same number of stars.
- Adjacent rows differ by no more than one star.
- The first row cannot have fewer stars than the second row.

Your team sees beyond the short-term change to 51 for the U.S. flag. You want to corner the market on flags for any union of three or more states. Given the number S of stars to draw on a flag, find all possible *visually appealing* flag patterns.

Input

The input consists of a single line containing the integer S ($3 \leq S \leq 32{,}767$).

Output

On the first line, print S, followed by a colon. Then, for each visually appealing flag of S stars, print its compact representation, one per line.

This list of compact representations should be printed in increasing order of the number of stars in the first row; if there are ties, print them in order of the number of stars in the second row. The cases 1-by-S and S-by-1 are trivial, so do not print those arrangements.

The compact representations must be printed in the form "`x,y`", with exactly one comma between `x` and `y` and no other characters.

Sample Input and Output

`3`	`3:` `2,1`

`50`	`50:`
	`2,1`
	`2,2`
	`3,2`
	`5,4`
	`5,5`
	`6,5`
	`10,10`
	`13,12`
	`17,16`
	`25,25`

`51`	`51:`
	`2,1`
	`3,3`
	`9,8`
	`17,17`
	`26,25`

Problem Y — 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 Z — limit 1 second

Forbidden Zero

You're writing the positive integers in increasing order starting from one. But you've never learned the digit zero, and thus omit any number that contains a zero in any position. The first ten integers you write are: 1, 2, 3, 4, 5, 6, 7, 8, 9, and 11.

You have just written down the integer n (which is guaranteed to not contain the digit zero). What will be the next integer that you write down?

Input

The input consists of a single line containing the integer n ($1 \leq n \leq 999{,}999$).

It is guaranteed that n does not contain the digit zero.

Output

Print, on a single line, the next integer you will be writing down.

Sample Input and Output

99	111

1234	1235