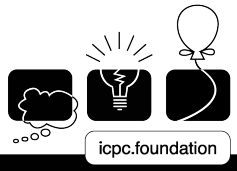


acm
icpc International Collegiate Programming Contest

icpc north america sponsor

icpc global programming tools sponsor

Problem Set

2017/2018 Southern California Regional

2017/2018 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 1
Latin Squares

A Latin Square is an $n \times n$ array filled with n different symbols, with each symbol 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 first (top) row and first (leftmost) column are in their natural order.

Your team is to write a program that will read a series of square $n \times n$ arrays, where n is in the range 2 through 36 inclusive. For each array your program is to determine if it is a Latin Square, and if so, if it is in reduced form.

Input to your program will be a series of square arrays. The first line for each array is the value of n, starting in the first column. The next n lines each contain n characters in base n, using the characters 0 through 9 and upper-case A (10) through Z (35). The last line of the last array is followed by end-of-file.

If the array is not a Latin Square, print a line containing only the string "No". If it is a Latin Square, but not in reduced form, print a line containing only the string "Not Reduced". If it is a Latin Square in reduced form, print a line containing only the string "Reduced". No leading or trailing whitespace is to appear on an output line.

Sample Input

```
3
012
120
201
4
3210
0123
2301
1032
11
0123458372A
A9287346283
0285475A834
84738299A02
1947584037A
65848430002
038955873A8
947530200A8
93484721084
95539A92828
04553883568
```

Problem 1
Latin Squares (continued)

Output for the Sample Input

```
Reduced
Not Reduced
No
```

2017/2018 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

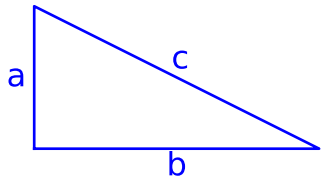
Problem 2
Congruent Numbers

A *congruent number* is an integer that is the area of some right triangle where the length of each side of the triangle is a rational number.

A rational number is a fraction, p/q, where p, the numerator, and q, the denominator, are integers. Note that if $q = 1$, then $p/1$ is an integer; therefore, an integer is a rational number.

The "congruent number problem" is: given an integer n, is it a congruent number? Mathematicians have been working on this since the Middle Ages, so far without success. There is a test that relies on the unproven Birch and Swinnerton-Dyer conjecture. If you can prove that conjecture you win a one million dollar prize from the Clay Mathematics Institute. (Your team can work on the proof after the contest.)

The task here is much easier: given a and b, the non-hypotenuse sides of a right triangle, determine if the area of that triangle is a congruent number. That is, determine if the lengths of all three sides of the triangle are rational numbers and the area n is an integer.



where $c^2 = a^2 + b^2$, $n = a \cdot b/2$, a, b, c are rational numbers and n is an integer.

Input to your program will be a series of lines terminated by end-of-file. Each line will consist of two positive rational numbers of the form p or p/q where p and q are integers with no embedded signs or spaces and the two rational numbers are separated by whitespace. The maximum number of digits in a numerator (p) or denominator (q) is 100.

For each input pair, your program is to print a line with n, the integer area of the triangle, if n is a congruent number or the word "no" if not. No leading or trailing whitespace is to be printed on an output line. There are to be no signs or leading zeroes in front of an integer.

Sample Input

```
3 4
3/2   20/3
3/5 4/5
1  10
335946000/2950969  233126551/167973000
20 21
12 35
65979511071975972/2305628412171265 16139398885198855/251830194931206
```

Problem 2
Congruent Numbers (continued)

Output for the Sample Input

```
6
5
no
no
79
210
210
917
```

2017/2018 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 3
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 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, as shown here:

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

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.

When displayed vertically, adjacent rows differ by no more than one star:

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

The compact representation for this is: 5,4.

A 51-star flag that preserves the relationship can have three rows of 9 stars, interlaced with three rows of 8 stars ($27 + 24 = 51$):

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

This compact representation is: 9,8.

A star arrangement 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, all the way up to 32,767. Your team is to write a program that will, given the number of stars to place on a flag (S), find all possible visually appealing star arrangements.

Problem 3
Star Arrangements (continued)

Input to your program is a list of values of S, one number per line starting in the first column, where $3 \leq S \leq 32{,}767$. For each line of input, your program is to print a line with the value of S immediately followed by a colon, then for each visually appealing star arrangement, print a line with a space followed by the compact star arrangement. Each compact star arrangement is to be printed in the form "`x,y`", with exactly one comma between `x` and `y` and no other characters.

The list of compact representations is to be printed in increasing order of the number of stars in the first row. If there are multiple compact representations with the same number of stars in the first row, print them in increasing order of the number of stars in the second row. The cases 1-by-S and S-by-1 are considered trivial, so do not print those arrangements.

Sample Input

```
3
50
51
```

Output for the Sample Input

```
3:
 2,1
50:
 2,1
 2,2
 3,2
 5,4
 5,5
 6,5
 10,10
 13,12
 17,16
 25,25
51:
 2,1
 3,3
 9,8
 17,17
 26,25
```

2017/2018 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 4
Halfway

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

Your friend wants to know when his program is halfway done. Assuming that all comparisons take the same amount of time, what will the last number printed be when the program is exactly halfway done? For an odd number of comparisons, this is when it's doing the middle comparison. For an even number, it's the first of the two middle comparisons. Note that since the earlier items have more comparisons than the later items, the answer is not simply $n/2$.

Input to your program will be a series of lines terminated by end-of-file. Each line will consist of a single integer n, ($2 \leq n \leq 10^9$), which is the number of items. The integer will have no sign and no leading or trailing spaces.

For each input value, your program is to print a line containing an integer with no sign or leading or trailing spaces that is the last number printed by your friend's program before it does the halfway comparison.

Sample Input

```
2
500
1919
1000000000
72
7
```

Output for the Sample Input

```
1
147
562
292893219
21
2
```

2017/2018 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 5
Rainbow Roads

The Transit Authority of Greater Podunk is planning its holiday decorations. They want to create an illuminated display of their light rail map in which each stretch of track between stations can be illuminated in one of several colors.

At periodic intervals, the controlling software will choose two stations at random and illuminate all of the segments connecting those two stations. By design, for any two stations on the Greater Podunk Railway, there is a unique path connecting the two.

For maximum color and cheer, the display designers want to avoid having two adjacent segments of track lighting up in the same color. They fear, however, that they may have deviated from this guideline in the process of building the display. One of them has gone so far as to propose a means of measuring just how far from that ideal they may have fallen.

You are given a tree with n nodes (stations), 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 edges adjacent to 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. (A simple path is a path that does not repeat any vertex or edge.)

Your team is to write a program that will find all the *good* nodes in the given tree.

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.

Your program should first print a line containing k, the number of good nodes. On the next k lines, print the indices of all good nodes in increasing numerical order, one per line. No leading or trailing whitespace or leading zeroes are to appear on an output line.

(Note that for the sample below, node 3 is good because all paths that have node 3 as an endpoint are rainbow. In particular, even though the path $3 \to 4 \to 5 \to 6$ has two edges of the same color (i.e. $3 \to 4$, $5 \to 6$), it is still rainbow because these edges are not adjacent.)

Sample Input

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

Problem 5
Rainbow Roads (continued)

Output for the Sample Input

```
4
3
4
5
6
```

2017/2018 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 6
Haiku

A haiku is a three-line poem in which the first and third lines contain five syllables each, and the second line contains seven syllables.

An example of a haiku is:
Blue Ridge mountain road.
Leaves, glowing in autumn sun,
fall in Virginia.

Write a program to examine a line of English text and and attempt to render it as a haiku. This will require counting the syllables in the words of the text, which should be done according to the following rules:

1. A word consists of a maximal string of alphabetic characters (upper and/or lower-case), followed by zero or more non-blank, non-alphabetic characters.
    a. Upper/lower case distinctions are ignored for the purpose of counting syllables, but must be retained in the final output.
    b. Non-alphabetic characters are ignored for the purpose of counting syllables, but must be retained in the final output.
2. The characters "A", "E", "I", "O", "U", and "Y" are vowels. All other alphabetic characters are consonants, with these exceptions:
    a. The character sequence "QU" is considered to be a single consonant.
    b. The letter "Y" is considered to be a consonant if it is immediately followed by one of the other vowels.
3. Every word has at least one syllable. For example, "Fly","I", and "Ssshhh!" are words of one syllable.
4. Each (maximal) string of one or more consonants with at least one vowel to either side indicates a division into separate syllables. For example, "strong" has one syllable, "stronger" has two, and "bookkeeper" has three. "player" has two syllables (because the "y", being followed by an "e", is considered a consonant). There are two exceptions to this rule:
    a. An "E" appearing as the last alphabetic character in a word is silent and should be ignored unless the next-to-last alphabetic character is an "L" and the character immediately before that is another consonant. For example, "ale" and "pale" have one syllable. "able" has two.
    b. An "ES" sequence at the end of the alphabetic sequence in a word does not add a syllable unless preceded by two or more consonants. For example, "ales" and "pales" have one syllable. "witches" and "verses" have two.

Input to your program will consist of a series of lines of text consisting of a sequence of one or more words (as defined above) separated by single spaces. The total line length will not exceed 200 characters.

If the words in a given input line can be divided into a haiku, then print the haiku as three lines of output. Each line should be left-justified. A single space should separate each pair of words within a line. Each word should appear exactly as it does in the input, preserving case and any terminal non-alphabetic characters. Do not split a word across multiple lines.

If the words in the input cannot be divided into a haiku, print the line of input with no changes.

Problem 6
Haiku (continued)

Sample Input

```
Blue Ridge mountain road. Leaves, glowing in autumn sun, fall in Virginia.
Who would know if we had too few syllables?
International contest- motivation high Programmers have fun!.
```

Output for the Sample Input

```
Blue Ridge mountain road.
Leaves, glowing in autumn sun,
fall in Virginia.
Who would know if we had too few syllables?
International
contest- motivation high
Programmers have fun!.
```

2017/2018 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 7
Law 11

The *Laws of the Game* describe the rules to soccer (internationally known as "football"). Of the seventeen enumerated laws, Offside as described by Law 11 is perhaps the most contentious. Law 11 takes barely more than a single page to describe. In short, offside consists of two conditions: the offside position, and a state of play that turns the offside position into an offense. To consistently enforce offside position at youth soccer matches (and generally ratchet down the poor sportsmanship of parents who consider themselves smarter than the referees, but too selfish to serve as referees themselves), your team has been contacted to implement a program for offside position determination.

The offside position may be stated as

a) *any part of the head, body, or feet (but not hands or arms) is in the opponents' side of the field (excluding the halfway line)*, AND

b) *any part of the head, body, or feet (but not hands or arms) is closer to the opponents' goal line than BOTH the ball and the second-to-last opponent.*

Your program is to treat all lines as one-dimensional, with the ball and players treated as points. Condition (a) is satisfied anytime the point representing a player is in the opponent's half of the field. (A player at mid-field ($x = 0$) is not in the opponent's half of the field.) Your program will unburden the referees from having to track 22 players spread across the playing field. This frees the referees to make the judgment calls that humans are better at.

Your program must take time-series data for the x, y position of the ball and the x, y positions of 22 players (11 on each team). Input to your program starts with a line describing the length, L, and width, W, of the soccer pitch, measured in meters. L and W are separated by a comma. The remaining input is a series of three-line groups. Within each group, the first line will contain the timestamp ("mm:ss", minutes and seconds into the half), followed by the x and y coordinates of the ball. These fields are separated by commas. The second line contains 22 comma-separated values, 11 x, y pairs representing the coordinates of players numbered 1 through 11 on team "Left", defending the goal on the left side of the field. The third line contains 22 comma-separated values as well; these are the x, y coordinates of players numbered 1 through 11 on team "Right", defending the goal on the right side of the field.

In Figure 1, team "Left" defends the goal at $x = -L/2$. Team "Right" defends the goal at $x = L/2$. The center of the field is at coordinate $(0, 0)$. Players stepping out of bounds are still considered in play; specifically, players beyond the goal lines ($|x| > L/2$) are considered to be at the respective goal line for offside position determination.

Only for time points (lines) that have players in offside position(s), print a line of output. Begin the line with the mm:ss timestamp of the data. If there are any players in offside positions for team Left, print a single space, followed by "Left". For each player on team Left that is in an offside position, print a single space followed by the number of that player, 1 through 11, in ascending order by player number. Similarly, if there are any players in offside positions for team Right, print a single space, followed by "Right". For each player on team Right that is in an offside position, print a single space followed by the number of that player, 1 through 11, in ascending order by player number.

Problem 7
Law 11 (continued)

Sample Input

```
90,50
00:00,0,25
-45,25,0,25,-10,20,-10,21,-10,22,-10,23,-10,24,-10,25,-10,26,-10,27,-10,28
45,25,10,20,10,21,10,22,10,23,10,24,10,25,10,26,10,27,10,28,10,29
00:01,-5,25
-45,25,15,25,-10,20,-10,21,-10,22,-10,23,-5,24,-10,25,-10,26,-10,27,-10,28
45,25,10,20,10,21,10,22,10,23,10,24,10,25,10,26,10,27,10,28,20,29
00:02,10,25
-45,25,20,25,-10,20,-10,21,-10,22,-10,23,-5,24,-10,25,-10,26,-10,27,-10,28
45,25,10,20,10,21,10,22,10,23,10,24,10,25,10,26,10,27,10,28,15,29
00:03,15,25
-45,25,20,25,-10,20,-10,21,0,22,-10,23,-5,24,-10,25,-10,26,-10,27,-10,28
45,25,10,20,10,21,10,22,10,23,10,24,10,25,10,26,10,27,10,28,15,25
00:04,0,25
-45,25,15,26,-10,20,-10,21,10,22,-10,23,-5,24,-10,25,-10,26,-10,27,-10,28
45,25,10,20,10,21,10,22,10,23,10,24,10,25,10,26,10,27,10,28,15,25
00:05,-5,24
-45,25,20,26,-10,20,-10,21,20,22,-10,23,-5,24,-10,25,-10,26,-10,27,-10,28
45,25,10,20,10,21,10,22,10,23,10,24,10,25,10,26,10,27,10,28,15,25
00:06,10,25
-45,25,15,26,-10,20,-10,21,-10,22,-10,23,-5,24,-10,25,-10,26,-10,27,-10,28
45,25,10,20,10,21,10,22,10,23,10,24,10,25,10,26,10,27,10,28,15,25
```

Output for the Sample Input

```
00:02 Left 2
00:03 Left 2
00:05 Left 2 5
```

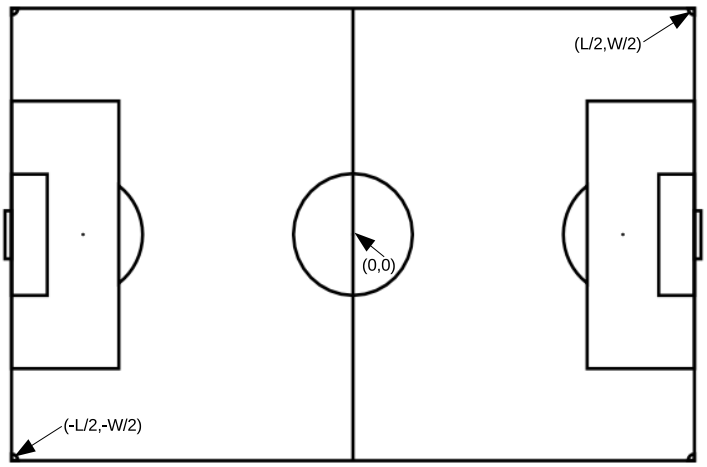


Figure 1. Soccer field (pitch) with coordinates.

2017/2018 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 8
Juggling for Nerds

Juggling for Nerds is a book in progress that teaches computer nerds how to juggle. The entire chapter on one-handed juggling simply states "One-handed juggling is a trivial matter of:

(a) choosing the number of identical objects to juggle,

(b) determining how fast you are capable of catching and tossing a single object,

(c) determining how fast you can reposition your hand to catch a single falling object,

(d) disregarding atmospheric drag,

(e) using a Newtonian model of Earth's gravity at sea level, and

(f) doing the math. There is ample space in the margin of this page to do the math."

Your program must implement instruction (f). Input to your program will be a series of lines with four values per line, separated by whitespace. The first value starts at the beginning of the line; it represents n, the number of identical objects to juggle. The second value is m, the mass of any one object, in kg. The third value is t_{ct}, the time in seconds the juggler can catch then toss the object (from the moment of contact with the falling object to the moment the object separates from the hand). The fourth value is t_r, the time in seconds the juggler takes to reposition to catch another object.

For each line of input, your program is to produce a single line of output specifying h_t, how high to throw each object to achieve the fastest possible juggling. h_t is measured from the "catch-release plane," an arbitrary plane above and parallel to the ground unique to each nuggler (nerd juggler). Express h_t in units of meters, rounded to two places after the decimal point. Do not put any leading or trailing whitespace on output lines.

Your Newtonian model of gravity states that

$$h = \frac{1}{2}gt^2$$

where h is the the distance an object falls from rest in t seconds. g is gravitational acceleration, $9.8m/s^2$.

There will be no more than 20 objects to be juggled. You may assume that the nuggler is strong enough to throw the objects to the required height.

Sample Input

```
1 0.4 0.5 0.3
2 0.6 0.45 0.1
```

Output for the Sample Input

```
0.11
0.52
```

2017/2018 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 9
Long Long Strings

DNA editing is done as a series of insert and delete operations. To store and edit DNA sequences your company has developed a LongLongString class that can store strings with up to $N = 10$ billion (10^{10}) alphanumeric characters. The class supports two basic operations:

Ins(p, c) - inserts the character c at position p, $(0 \leq p \leq \text{length(string)})$

Del(p) - deletes the character at position p, $(0 \leq p < \text{length(string)})$

For insertion, $p = 0$ indicates insertion at the beginning of the DNA string, and $p = \text{length(string)}$ indicates appending at the end of the string.

Your job is to write a program that compares two DNA editing *programs* to determine if their effects are identical—that is, if the editing programs are applied independently to any sufficiently long string, they would result in the same string. The programs are of size L_1 and L_2, $(0 \leq L_i < 10000)$. For example:

Del(1) Del(2) and Del(3) Del(1) are identical.

Del(2) Del(1) and Del(1) Del(2) are different.

Ins(1,x) Del(1) and the empty program 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 to your program is a series of DNA editing comparisons. Each comparison begins with a line containing the integer N, the initial length of the DNA sequence $(0 \leq N \leq 10^{10})$. The following lines contain two DNA editing programs, with each program ended by a line with only the letter 'E'. Each DNA program is zero or more operations. Any operations are one per line of the form

I p c - Insert character c at position p. c is an alphanumeric character.

D p - Delete character at position p.

For each DNA editing comparison, print a line containing only the string "Identical" if the two programs produce the same resulting string, or only the string "Different" if the two programs produce different strings.

Problem 9
Long Long Strings (continued)

Sample Input

```
10
D 1
D 2
E
D 3
D 1
E
5
D 2
D 1
E
D 1
D 2
E
31
I 1 x
D 1
E
E
255
I 14 b
I 14 a
E
I 14 a
I 15 b
E
255
I 14 a
I 15 b
E
I 14 b
I 15 a
E
```

Output for the Sample Input

```
Identical
Different
Identical
Identical
Different
```

2017/2018 SOUTHERN CALIFORNIA REGIONAL ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 10
Matrix Multiplication

Winning an ICPC regional has expanded your employment opportunities. The keywords "ICPC Finalist" have been flagged by the resume filters set up by *Numerical Concoctions, Inc.*, an early stage startup aiming to disrupt the market for cloud-based linear algebra services. As a result you have been offered a position as a "Matrix Ninja Intern", working on the next generation matrix multiplication library functions.

When it comes to matrix multiplications, the company has already solved the problem for $N = 2$. Their function $\mathrm{Mul}(A, B)$ for multiplying two matrices is dominating all benchmarks. The company's next goal is to be the best at multiplying N matrices in a row for any N. The plan is to leverage the existing technology and simply call $\mathrm{Mul}(A, B)$ $N - 1$ times.

As you recall from Linear Algebra 101, the product of two matrices $A[n, m]$, and $B[m, l]$, is a matrix $X[n, l]$. Things become more interesting with three or more matrices, because even though matrix multiplication is associative, that is, $(A \times B) \times C = A \times (B \times C)$, different ways of computing the final product require different amounts of intermediate storage. For example, if we have $A[1000, 2]$, $B[2, 1000]$, and $C[1000, 2]$, computing $A \times B$ first requires a matrix with $1000 \times 1000 = 1000000$ elements to be allocated, whereas computing $B \times C$ first requires a matrix with only $2 \times 2 = 4$ elements to be allocated.

Your first assignment at your new job is to implement a helper function that, given the sizes of the N matrices to be multiplied together, determines how to compute the product allocating as little intermediate storage as possible. Assume that all intermediate storage cannot be reused and that it will be deallocated only after the final product is computed.

For this problem, the input will consist of lines describing individual test cases in the $[n_1, m_1] * [n_2, m_2] * \ldots * [n_N, m_N]$ format, where $2 \le N \le 1000$, $1 \le m_i \le 10000$, and $1 \le n_i \le 10000$.

For each input line, your program is to print the number of elements that need to be allocated as intermediate storage. Your program is not to count the input matrices or the final product as intermediate storage. No leading or trailing whitespace or leading zeroes or signs are to appear on output lines.

Sample Input

```
[1000,2]*[2,1000]*[1000,2]
[20,30]*[30,20]
[2,1]*[1,1]*[1,1]*[1,1]*[1,2]
[22,22]*[22,55]*[55,76]*[76,29]
```

Output for the Sample Input

```
4
0
4
2233
```