Problem A. Assignment Algorithm

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

A low-budget airline is designing a sophisticated algorithm that will assign more desirable seats to passengers who buy tickets earlier. Their airplane has r rows of seats, where r is an even integer. There are also 3 *exit rows* in the airplane; those rows do not contain any seats but only provide access to the emergency exits. One exit row is in the very front of the airplane (before the first row of seats), one in the very back (behind the last row of seats) and one right in the middle. The rows are numbered with integers 1 through $r + 3$ with row numbers increasing from the front to the back of the airplane. Rows numbered 1, $r/2 + 2$ and $r + 3$ are exit rows while all the other rows are seat rows.

The seating configuration is "3–3–3" — each seat row contains three groups of three seats with the passenger aisles between the groups. Seats in the same row are denoted with consecutive letters left to right corresponding to the pattern "`ABC.DEF.GHI`".

When a passenger purchases a ticket, she is assigned a seat according to the following rules:

1. If there is an empty seat in a row directly after an exit row, all other rows are ignored in the following step (but they are not ignored when balancing the airplane in the last step).
2. First, we select a seat row with the largest number of empty seats. If there are multiple such rows, we select the one closest to an exit row (distance between rows a and b is simply $|a - b|$). If there are still multiple such rows, we select the one with the lowest number.
3. Now, we consider empty seats in the selected row and select one with the highest *priority*. Seat priorities, from highest to lowest are as follows:
   (a) Aisle seats in the middle group (`D` or `F`).
   (b) Aisle seats in the first and third group (`C` or `G`).
   (c) Window seats (`A` or `I`).
   (d) Middle seat in the middle group (`E`).
   (e) Other middle seats (`B` or `H`).

   If there are two empty seats with the same highest priority, we consider the balance of the entire airplane. The airplane's *left side* contains all seats with letters `A`, `B`, `C` or `D`, while the *right side* contains all seats with letters `F`, `G`, `H` or `I`. We select an empty seat in the side with more empty seats. If both sides have the same number of empty seats, we select the seat in the left side of the airplane.

Some of the airplane's seats are already reserved (possibly using a completely different procedure from the one described above). Determine the seats assigned to the next n passengers purchasing a ticket.

Input

The first line contains two integers r and n ($2 \leq r \leq 50$, $1 \leq n \leq 26$) — the number of seat rows in the airplane (always an even integer) and the number of new passengers purchasing tickets. The following $r + 3$ lines contain the current layout of the airplane. The j-th line contains exactly 11 characters — the layout of the row j of the airplane. Exit rows and aisles are denoted with the "`.`" characters. The "`#`" character denotes a reserved seat, while the "`-`" character denotes a seat that is currently empty. You may assume there will be at least n empty seats in the airplane.

Output

Output $r + 3$ lines containing the final layout of the airplane. The layout should be exactly the same as in input with the following exception: the seat assigned to the j-th passenger purchasing a ticket should be denoted with the j-th lowercase letter of the English alphabet.

Examples

standard input	standard output
`2 17`	`...........`
`...........`	`hnd.#lb.fpj`
`---.#--.---`	`...........`
`...........`	`kqg.cma.eoi`
`---.---.---`	`...........`
`...........`	
`6 26`	`...........`
`...........`	`gke.aic.###`
`---.---.###`	`#-#.mzo.r-v`
`#-#.---.---`	`x-p.###.n-t`
`---.###.---`	`...........`
`...........`	`fjb.###.dlh`
`---.###.---`	`#-s.#-#.w-#`
`#--.#-#.--#`	`#-u.qy#.#-#`
`#--.--#.#-#`	`...........`
`...........`	

Problem F. Faulty Factorial

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

The *factorial* of a natural number is the product of all positive integers less than or equal to it. For example, the factorial of 4 is $1 \cdot 2 \cdot 3 \cdot 4 = 24$. A *faulty factorial* of length n is similar to the factorial of n, but it contains a fault: one of the integers is *strictly smaller* than what it should be (but still at least 1). For example, $1 \cdot 2 \cdot 2 \cdot 4 = 16$ is a faulty factorial of length 4.

Given the length n, a *prime* modulus p and a target remainder r, find some faulty factorial of length n that gives the remainder r when divided by p.

Input

The first line contains three integers n, p and r ($2 \le n \le 10^{18}$, $2 \le p < 10^7$, $0 \le r < p$) — the length of the faulty factorial, the prime modulus and the target remainder as described in the problem statement.

Output

If there is no faulty factorial satisfying the requirements output “-1 -1”. Otherwise, output two integers — the index k of the fault ($2 \le k \le n$) and the value v at that index ($1 \le v < k$). If there are multiple solutions, output any of them.

Examples

standard input	standard output
`4 5 1`	`3 2`
`4 127 24`	`-1 -1`

Problem G. Gambling Guide

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

A railroad network in a nearby country consists of n cities numbered 1 through n, and m two-way railroad tracks each connecting two different cities. Tickets can only be purchased at automated machines installed at every city. Unfortunately, hackers have tampered with the ticket machines and now they all work as follows: when a single coin is inserted in the machine installed at city a, the machine dispenses a single one-way ticket from a to a *random* neighboring city. More precisely, the destination city is chosen uniformly at random among all cities directly connected to a with a railroad track. Destinations on different tickets originating in the same city are independent.

A computer science student needs to travel from city 1 (where she lives) to city n (where a regional programming contest has already started). She knows how the machines work (but of course cannot predict the random choices) and has a map of the railway network. In each city, when she purchases a ticket, she can either immediately use it and travel to the destination city on the ticket, or discard the ticket and purchase a new one. She can keep purchasing tickets indefinitely. The trip is finished as soon as she reaches city n.

After doing some calculations, she has devised a traveling strategy with the following properties:

- The probability that the trip will eventually finish is 1.
- The expected number of coins spent on the trip is the smallest possible.

Find the expected number of coins she will spend on the trip.

Input

The first line contains two integers n and m ($1 \le n, m \le 300\,000$) — the number of cities and the number of railroad tracks. Each of the following m lines contains two different integers a and b ($1 \le a, b \le n$) which describe a railroad track connecting cities a and b. There will be at most one railroad track between each pair of cities. It will be possible to reach city n starting from city 1.

Output

Output a single number — the expected number of coins spent. The solution will be accepted if the absolute or the relative difference from the judges solution is less than 10^{-6}.

Example

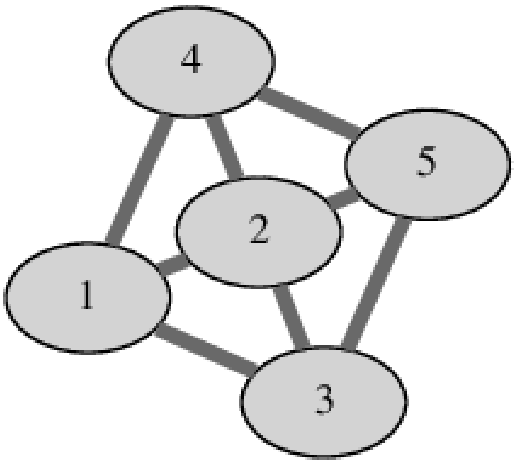
standard input	standard output

4 4
1 2
1 3
2 4
3 4
	3.0000000000

5 8
1 2
1 3
1 4
2 3
2 4
3 5
5 4
2 5
	4.1111111111

Note

Picture to Sample 2:



Problem H. Hidden Hierarchy

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are working on the user interface for a simple text-based file explorer. One of your tasks is to build a navigation pane displaying the *directory hierarchy*. As usual, the filesystem consists of directories which may contain files and other directories, which may, in turn, again contain files and other directories etc. Hence, the directories form a hierarchical tree structure. The top-most directory in the hierarchy is called the *root* directory. If directory d directly contains directory e we will say that d is the *parent directory* of e while e is a *subdirectory* od d. Each file has a *size* expressed in bytes. The directory size is simply the total size of all files directly or indirectly contained inside that directory.

All files and all directories except the root directory have a *name* — a string that always starts with a letter and consists of only lowercase letters and "`.`" (dot) characters. All items (files and directories) directly inside the same parent directory must have unique names. Each item (file and directory) can be uniquely described by its *path* — a string built according to the following rules:

- Path of the root directory is simply "`/`" (forward slash).
- For a directory d, its path is obtained by concatenating the directory names top to bottom along the hierarchy from the root directory to d, preceding each name with the "`/`" character and placing another "`/`" character at the end of the path.
- For a file f, its path is the concatenation of the parent directory path and the name of file f.

We display the directory hierarchy by *printing* the root directory. We print a directory d by outputting a line of the form "$m_d \text{␣} p_d \text{␣} s_d$" where p_d and s_d are the path and size of directory d respectively, while m_d is its *expansion marker* explained shortly. If d contains other directories we must choose either to *collapse* it or to *expand* it. If we choose to expand d we print (using the same rules) all of its subdirectories in lexicographical order by name. If we choose to collapse directory d, we simply ignore its contents.

The expansion marker m_d is a single blank character when d does not have any subdirectories, "`+`" (plus) character when we choose to collapse d or a "`-`" (minus) character when we choose expand d.

Given a list of files in the filesystem and a threshold integer t, display the directory hierarchy ensuring that each directory of size at least t is printed. Additionally, the total number of directories printed should be minimal. Assume there are no empty directories in the filesystem — the entire hierarchy can be deduced from the provided file paths. Note that the root directory has to be printed regardless of its size. Also note that a directory of size at least t only has to be *printed*, but not necessarily *expanded*.

Input

The first line contains an integer n ($1 \le n \le 1\,000$) — the number of files. Each of the following n lines contains a string f and an integer s ($1 \le s \le 10^6$) — the path and the size of a single file. Each path is at most 100 characters long and is a valid file path according to the rules above. All paths will be different.

The following line contains an integer t ($1 \le t \le 10^9$) — the threshold directory size.

Output

Output the minimal display of the filesystem hierarchy for the given threshold as described above.

Example

standard input	standard output
9 /sys/kernel/notes 100 /cerc/problems/a/testdata/in 1000000 /cerc/problems/a/testdata/out 8 /cerc/problems/a/luka.cc 500 /cerc/problems/a/zuza.cc 5000 /cerc/problems/b/testdata/in 15 /cerc/problems/b/testdata/out 4 /cerc/problems/b/kale.cc 100 /cerc/documents/rules.pdf 4000 10000	- / 1009727 - /cerc/ 1009627 /cerc/documents/ 4000 - /cerc/problems/ 1005627 - /cerc/problems/a/ 1005508 /cerc/problems/a/testdata/ 1000008 + /cerc/problems/b/ 119 + /sys/ 100
8 /b/test/in.a 100 /b/test/in.b 1 /c/test/in.a 100 /c/test/in.b 1 /c/test/pic/in.a.svg 10 /c/test/pic/in.b.svg 10 /a/test/in.a 99 /a/test/in.b 1 101	- / 322 + /a/ 100 - /b/ 101 /b/test/ 101 - /c/ 121 + /c/test/ 121
2 /a/a/a 100 /b.txt 99 200	+ / 199

Problem J. Justified Jungle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

As you probably know, a *tree* is a graph consisting of n nodes and $n - 1$ undirected edges in which any two nodes are connected by exactly one path. A *forest* is a graph consisting of one or more trees. In other words, a graph is a forest if every connected component is a tree. A forest is *justified* if all connected components have the same number of nodes.

Given a tree G consisting of n nodes, find all positive integers k such that a justified forest can be obtained by erasing exactly k edges from G. Note that erasing an edge never erases any nodes. In particular when we erase all $n - 1$ edges from G, we obtain a justified forest consisting of n one-node components.

Input

The first line contains an integer n ($2 \le n \le 1\,000\,000$) — the number of nodes in G. The k-th of the following $n - 1$ lines contains two different integers a_k and b_k ($1 \le a_k, b_k \le n$) — the endpoints of the k-th edge.

Output

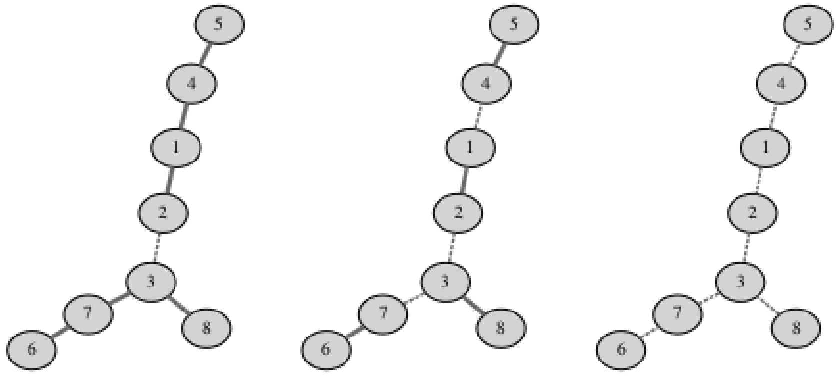
The first line should contain all wanted integers k, in increasing order.

Example

standard input	standard output

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

Note



Figures depict justified forests obtained by erasing 1, 3 and 7 edges from the tree in the example input.

Problem L. Lunar Landscape

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A satellite is surveying a possible rover landing area on the moon. The landing area is modeled as a square grid embedded in the standard coordinate system.

The satellite has taken n photos, each capturing a square area of the surface. Careful camera calibration has ensured that all photos are aligned with the grid — all four vertices have integer coordinates. Due to the satellite's changing orbit there are two types of photos:

- Photos of type `A` have sides that are parallel to coordinate axes. Such a photo is specified by giving the integer coordinates (x, y) of the square's middle point and the length of its side a — always an even integer.
- Photos of type `B` have sides at a 45° angle to the coordinate axes. Such a photo is specified by giving the integer coordinates (x, y) of the square's middle point and the length of its diagonal d — always an even integer.

Find the total surface area captured in the satellite photos.

Input

The first line contains an integer n ($1 \le n \le 200\,000$) — the number of photos. The j-th of the following n lines is either of the form "`A` x_j y_j a_j" or "`B` x_j y_j d_j" representing a photo of type `A` or `B`, respectively. The x_j and y_j are the integer coordinates of the middle point of the photo ($-1\,000 \le x_j, y_j \le 1\,000$). The a_j and d_j are even integers ($2 \le a_j, d_j \le 1\,000$) — the side length and the diagonal length, respectively.

Output

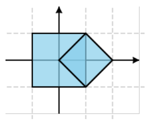
Output a number with **exactly two** digits after the decimal point — the total area of the surface. The answer has to exactly correspond to the judge's solution (no rounding errors are tolerated).

Example

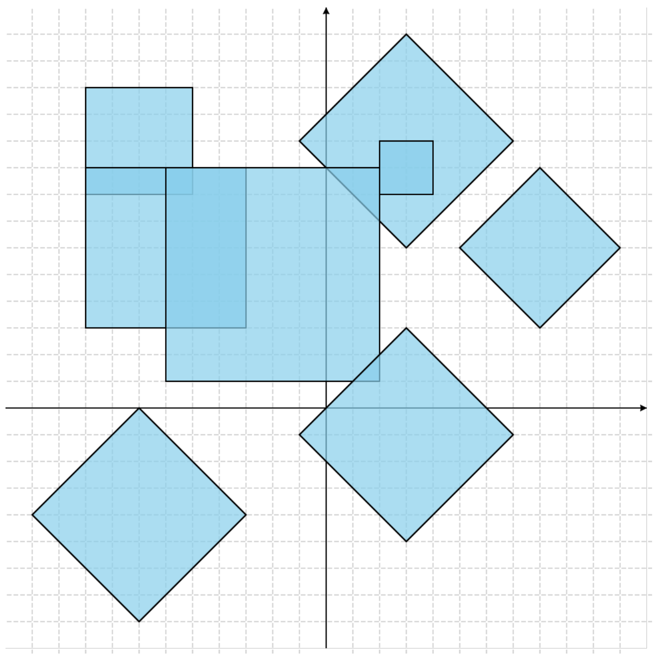
standard input	standard output
2 A 0 0 2 B 1 0 2	5.00
8 A -7 10 4 B 3 10 8 A -6 6 6 A -2 5 8 B 3 -1 8 B -7 -4 8 A 3 9 2 B 8 6 6	205.50

Note

Sample 1



Sample 2



Problem M. Mobile Management

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You have a single robot, and would like to use it to produce n mobile phones. However, assembling the phones one by one for one robot takes a long time, so it may be more time-efficient to first use the robot to assemble a new robot. This new robot may then in turn be used to assemble mobile phones or even more robots. Assembling jobs take a full hour, and every hiur you can choose for each robot in your possession to have it assemble a phone or to have it to assemble a new robot (which becomes available for use the next hour).

What is the minimum possible number of hours needed to build at least n mobile phones?

Input

The input contains a single integer n ($1 \le n \le 10\,000$), the number of mobile phones you need to assemble.

Output

Output a single integer, the minimum number of hours needed to assemble at least n mobile phones.

Example

standard input	standard output
1	1
5	4

Problem N. Natural Numbers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Given positive integer N, find four pairwise distinct positive integers A, B, C, D such as $A + B = C \times D = N$.

Input

Input consists of one integer N ($5 \le N \le 1042$).

Output

Print four distinct positive integers A, B, C, D such as $A + B = C \times D = N$. If there are more than one solutions, print any.

Examples

standard input	standard output
15	7 8 3 5
42	4 38 3 14

Problem O. OCR Outsorcing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 Mebibytes

Outsourcers created new module of Optical Character Recognition (OCR) for the new smartphone of company A. Such a modules are used for reading paper documentation. Obviously, the transformation process is not 100% reliable and some characters are not recognized. Your task is to write a program that determines the recognition process efficiency ratio. The ratio should be computed as R/A where R is the number of recognized characters and A is the number of all characters. Newlines do not count as characters.

Input

Input file contains at least one line of processed text where unrecognized characters are represented by "#". No line will be longer than 100 characters. Line may contain only symbols with ASCII-codes from 32 to 126.

Output

Print the ratio in percent, rounded to the nearest number with no more than one digit after the decimal point (0.05 rounds up to 0.1). Do not print extra zeroes after decimal point (i.e. 0.50 instead of 0.5 or 25.0 instead of 25 are considered as wrong answer).

Example

standard input	standard output

`Pr#nt the r#tio in perce##,`
`rounde##to t#e nearest number`
	`87.7`

`The Clear Text.`	`100`

Problem P. Production Planning

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 Mebibytes

The company A is ready to start his new project — production of new model of the smartphone. The project is quite expensive, so the company can't afford it without investors.

The management of the company also knows how many companies would invest some money in this project and how many each would invest. But they like to use as few investors as possible and if project demands too many money then they rather not start it at all.

Can you tell him the minimum number of companies A. needs to take money from?

Input

The first line contains one integer — number of test cases T ($1 \le T \le 255$). Each test case describes one project and its first line contains 2 integers: amount of money N ($1 \le N \le 10^6$) A. needs to product first serie of smartphones and number of companies C ($1 \le C \le 1000$) which are ready to invest some money in project.

In a second line youll get C integers, i-th of them is amount of money M_i — maximal investiion of i-th company ($1 \le M_i \le 10^4$).

Output

For each test case print a single line with the minimum number of investors the company needs to start his project. If its impossible even if he will use maximal investitions of all investors, write "`impossible`".

Examples

standard input	standard output

3
101 6
23 27 41 19 23 57
100 7
20 22 33 8 24 67 9
1001 3
123 456 123
	3
	2
	impossible

Problem Q. Quiz Questions

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 Mebibytes

The mobile games company plans to create the quiz game for the phones without screen autorotation. The killing feature is that words and questions will be in some way resistant to the screen rotation. But first they need to evaluate the *rotability* of the words

Rotating the shape of lowercase letters 180 degrees can transform them into other characters or into themselves. Such symmetric characters can be used in words, which when rotated, form the same word, such as "sos" and "mow". Parameter "rotability" $Rr(w)$ scores word w with respect to how symmetric it is when given a 180-degree spin.

The letters that are symmetric with themselves are '`o`', '`l`', '`s`' and '`x`'.

The pairs of letters that are symmetric, are '`b`' and '`q`', '`d`' and '`p`', '`m`' and '`w`', '`n`' and. '`u`.

The score $s(c_1, c_2)$ of two letters c_1 and c_2 is 3 if the two letters are the same and symmetric, 2 if the letters are different and form a symmetric pair, 1 if the letters are the same but not symmetric, and 0 if the letters are not the same and not symmetric.

The rotability $Rr(w)$ of a n-letter word w is the sum of the scores of the letter pairs: $Sp(w) = s(w_1, w_n) + s(w_2, w_{n-1}) + \ldots + s(w_n, w_1)$.

Your task is to calculate rotation range of some words from the quiz questions.

Input

Input file contains a list of 100 or less lowercase alphabetic words, one per line. Each word contain at least one and at most 10 letters.

Output

For every word in the input, print at new line the word itself and its rotability. Print words in order they appear in input file.

Example

standard input	standard output
spellers ololo meow axe pad sos nu	spellers 14 ololo 15 meow 4 axe 3 pad 5 sos 9 nu 4

Problem R. Restore Radius

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 Mebibytes

There is a large circle with radius R and n small circles with radius r that are placed inside on the border of the large circle. Each small circle touch large circle and two (one, if $n = 2$) other small circles. Your job is, given R and n, to restore r.

Input

First line of input file contains one integer T ($1 \le T \le 4100$) — number of test cases. Each test case consists of one line containing a float R (not more than six digits after decimal point) and an integer n ($1 \le R \le 100$, $2 \le n \le 100$).

Output

For each test case print in new line radius r of small circle with absolute error 10^{-3} or less.

Example

standard input	standard output

4
5.0 7
5.0 19
3.1415 20
56 2
	1.513
	0.707
	0.425
	28.000