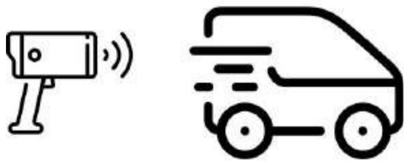
Problem A. Speeding

Source file name:	speeding.c, speeding.cpp, speeding.java, speeding.py
Input:	`Standard`
Output:	`Standard`



You'd like to figure out whether a car was speeding while it was driving down a straight road. Unfortunately you don't have any radar guns or related instruments for measuring speed directly; all you have are photographs taken of the car at various checkpoints on the road at various times. Given when and where these photographs were taken, what is the fastest speed that you can prove the car must have been going at some point along the road?

Input

The first line contains an integer N, the number of photographs taken, with $2 \leq N \leq 100$. The following N lines each contain two integers t_i and d_i, with $0 \leq t_i \leq 10000$ and $0 \leq d_i \leq 1000000$. The first photograph is always taken at time 0 with distance 0. Both the times and distances strictly increase. That is, $t_{i+1} > t_i$ and $d_{i+1} > d_i$

Output

Output the greatest integral speed that you can be certain the car was going at some point.

Example

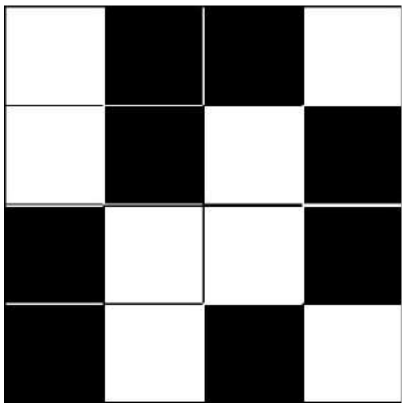
Input	Output

2
0 0
7 42
	6

5
0 0
5 24
10 98
15 222
20 396
	34

Problem B. Black and White

Source file name:	blackandwhite.c, blackandwhite.cpp, blackandwhite.java, blackandwhite.py
Input:	`Standard`
Output:	`Standard`



You are given an n-by-n grid where each square is colored either black or white. A grid is *correct* if all of the following conditions are satisfied:

- Every row has the same number of black squares as it has white squares.
- Every column has the same number of black squares as it has white squares.
- No row or column has 3 or more consecutive squares of the same color.

Given a grid, determine whether it is *correct*.

Input

The first line contains an integer n ($2 \leq n \leq 24$; n is even). Each of the next n lines contains a string of length n consisting solely of the characters '`B`' and '`W`', representing the colors of the grid squares.

Output

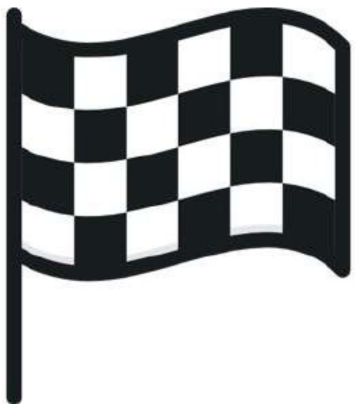
If the grid is *correct*, print the number 1 on a single line. Otherwise, print the number 0 on a single line.

Example

Input	Output
4 WBBW WBWB BWWB BWBW	1
4 BWWB BWBB WBBW WBWW	0
6 BWBWWB WBWBWB WBBWBW BBWBWW BWWBBW WWBWBB	0
6 WWBBWB BBWWBW WBWBWB BWBWBW BWBBWW WBWWBB	1

Problem C. Checkerboard

Source file name:	checkerboard.c, checkerboard.cpp, checkerboard.java, checkerboard.py
Input:	`Standard`
Output:	`Standard`



An R-by-C grid of squares is to be colored in a checkerboard style. The board will be filled with rectangles made up of the grid squares. There should be A rectangles vertically and B rectangles horizontally. All rectangles in row i should be a_i squares high; all rectangles in column j should be b_j squares wide. The top-left rectangle should be black and two adjacent rectangles that share a side should be colored differently.

Print the checkerboard.

Input

The first line will contain four integers, R, C, A, and B with $1 \le A \le R \le 50$ and $1 \le B \le C \le 50$. The next A lines will each contain a single positive integer a_i; the sum of the a_i values will be R. The next B lines will each contain a single positive integer b_i; the sum of the b_i values will be C

Output

Print the required checkerboard. It should have R lines each with a string of length C containing only the characters '`B`' and '`W`'

Example

Input	Output

`6 5 3 2`
`1`
`2`
`3`
`3`
`2`
	`BBBWW`
	`WWWBB`
	`WWWBB`
	`BBBWW`
	`BBBWW`
	`BBBWW`

`4 4 2 2`
`1`
`3`
`3`
`1`
	`BBBW`
	`WWWB`
	`WWWB`
	`WWWB`

Problem D. Pea Soup and Pancakes

Source file name:	peasoup.c, peasoup.cpp, peasoup.java, peasoup.py
Input:	`Standard`
Output:	`Standard`

As a Swede, you hold a deep love for the traditional Thursday lunch of pea soup and pancakes. You love it so much, in fact, that you will eat it any meal it is available. You find yourself looking at the menus for all your favorite restaurants every day to see if this combination is available, and realized you can do this more easily with a program. Given a list of restaurant menus, decide where to eat.

Input

The first line of input contains a number n ($1 \leq n \leq 10$), the number of restaurants. Then follow the n restaurant menus. Each menu starts with a line containing a number k ($1 \leq k \leq 10$), the number of menu items for the day. The remainder of the menu consists of $k + 1$ lines, each containing a nonempty string of at most 100 characters. The first of these lines is the restaurant name, and the rest are menu items. Strings consist only of lower case letters '`a`'-'`z`' and spaces, and they always start and end with a letter. All restaurant names are unique.

Output

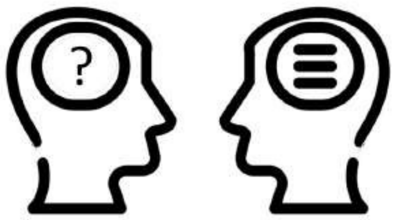
Output a single line. If at least one restaurant has both "`pea soup`" and "`pancakes`" as menu items, output the name of the first of those restaurants, by the order in which the restaurants appear in the input. Otherwise, output "`Anywhere is fine I guess`".

Example

Input	Output
2 2 q potatoes salad 3 nymble pancakes pea soup punsch	nymble
4 2 asian wok house paa soup pancakes 2 kebab kitchen pea soup pancakes 2 la campus tasty pea soup pancakes 3 slime stand slime pea soup and pancakes slime	Anywhere is fine I guess

Problem E. Even or Odd

Source file name:	evenorodd.c, evenorodd.cpp, evenorodd.java, evenorodd.py
Input:	`Standard`
Output:	`Standard`



Your friend has secretly picked N consecutive positive integers between 1 and 100, and wants you to guess if their sum is even or odd.

If the sum must be even, output '`Even`'. If the sum must be odd, output '`Odd`'. If the sum could be even or could be odd, output '`Either`'.

Input

The input is a single integer N with $1 \leq N \leq 10$.

Output

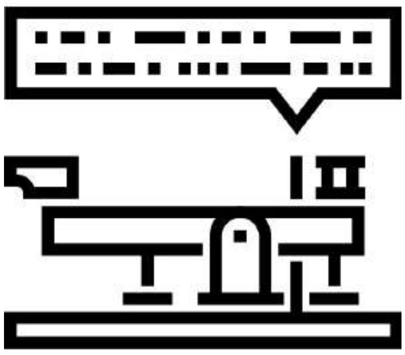
Output a single word. The word should be '`Even`', '`Odd`', or '`Either`', according to the rules given earlier.

Example

Input	Output
1	Either
2	Odd

Problem F. Remorse

Source file name:	remorse.c, remorse.cpp, remorse.java, remorse.py
Input:	`Standard`
Output:	`Standard`



A Morse-like code is an assignment of sequences of dots and dashes to alphabet characters. You are to create a Morse-like code that yields the shortest total length to a given message, and return that total length.

A dot has length 1. A dash has length 3. The gap between dots and dashes within a character has length 1. The gap between characters has length three. Spaces, punctuation, and alphabetic case are ignored. For example, the text

`The quick brown dog jumps over the lazy fox.`

is encoded as though it were just

`THEQUICKBROWNDOGJUMPSOVERTHELAZYFOX`

For example, with input `ICPC`, the answer is 17: Encode the `C`'s with a single dot, the `I` with a dash, and the `P` with two dots, for a total of '- . .. .' which has length $3 + 3 + 1 + 3 + 1 + 1 + 1 + 3 + 1$ or 17.

Input

The input will be a single line consisting of uppercase or lowercase letters, spaces, commas, periods, exclamation points, and question marks. The line will have a maximum length of 32 000 characters and will contain at least one letter. Everything but the letters should be ignored.

Output

The output will consist of the length of the encoded string when an optimal Morse-like code is used.

Example

Input	Output
`ICPC`	`17`
`A`	`1`
`The quick brown dog jumps over the lazy fox.`	`335`

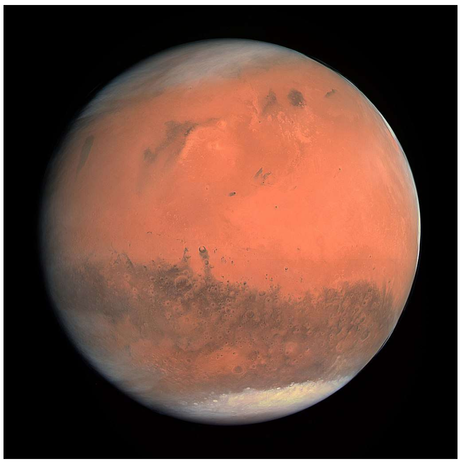
Problem G. Mars Window

Source file name:	marswindow.c, marswindow.cpp, marswindow.java, marswindow.py
Input:	`Standard`
Output:	`Standard`

You are an eccentric billionaire with an affinity for sending stuff to Mars. In an ideal world you would send stuff to Mars all the time, but your engineers tell you that it is most efficient to do it when Earth and Mars are somewhat close to each other.

Your engineers have calculated that optimal launch windows occur once every 26 months, and that one of them occurs in April 2018. They also tell you that they will not have any Big Finished Rockets by then, so you will have to wait for a later launch window.

Since your rocket scientists apparently can not be bothered to tell you about the optimal launch windows before it is too late, you have to keep track of that yourself. Write a program that determines if there is an optimal launch window in any given year.


Picture by the European Space Agency, cc by-sa

Input

The only line of input contains an integer y ($2018 \leq y \leq 10000$), the year you are interested in.

Output

Output "`yes`" if there is an optimal launch window in the year y, otherwise output "`no`".

Example

Input	Output
2018	yes
2019	no
2020	yes
2028	no

Problem H. Pairing Socks

Source file name:	pairingsocks.c, pairingsocks.cpp, pairingsocks.java, pairingsocks.py
Input:	`Standard`
Output:	`Standard`

Simone's mother often complains about how Simone never helps with chores at home. In return, Simone often points out that many of the chores her mother assigns her are NP-complete to perform optimally (like cleaning the house, seating her little brothers around the dinner table in a conflict-free way, splitting the brothers' Halloween loot in a fair manner and so on).

Being a computer scientist, her mother finds this a fair objection. Looking over her list of potential chores, she picked one she thinks should be easy to solve – pairing a number of different kinds of socks.

In the beginning, there are $2n$ socks stacked in a pile. To pair the socks, Simone can repeatedly make one of three moves:

1. Move the sock from the top of the original pile to the top of an auxiliary pile (which is originally empty).
2. Move the sock from the top of the auxiliary pile to the top of the original pile.
3. Pair the top socks from each pile together, if they are of the same type.

Picture by Villy Fink Isaksen, cc by-sa

Simone only has one auxiliary pile, for a total of two piles. There may be more than two socks of each type. In this case, Simone can pair them up however she wants.

Your task is to help Simone to determine the least number of moves she needs to pair the socks, if it is possible at all.

Input

The first line of input contains the integer n ($1 \leq n \leq 10^5$) as described above. The next line contains $2n$ integers $a_1, \ldots, a_{2n}$ ($1 \leq a_i \leq 10^9$ for each i), where a_i denotes the type of sock number i. Initially, sock 1 is at the top of the pile and sock $2n$ is at the bottom.

Output

If Simone can pair all the socks, output the least number of moves she needs to do this. If it is impossible to do so, output "`impossible`" (without the quotes).

Example

Input	Output

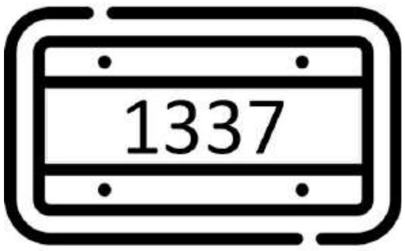
`2`
`1 2 2 1`
	`4`

`1`
`3 7`
	`impossible`

`3`
`5 5 5 5 5 5`
	`6`

Problem I. Issuing Plates

Source file name:	plates.c, plates.cpp, plates.java, plates.py
Input:	`Standard`
Output:	`Standard`



You are writing the code to check license plates. These consist of upper letters 'A'-'Z' and numbers '0'-'9'. You want to make sure the codes do not contain any bad words, even considering leetspeak.

Given some input strings, which are valid license plates?

In leetspeak we have the following equivalences: `0=O 1=L 2=Z 3=E 5=S 6=B 7=T 8=B`

Input

The first line will contain the integers N and M with $0 \leq N, M \leq 100$. Following this will be N lines containing bad words; each such word will contain only uppercase alphabetic characters ('A'-'Z') and be at most 25 characters long. Then there will be M lines containing the plates to check; each such plate will consist of only uppercase alphabetic characters and numeric digits ('0'-'9') and be at most 25 characters long.

Output

Your output will be M lines, one per plate, in the same order as the plates are given on the input. If the plate is valid, write out the string 'VALID'; otherwise write out the string 'INVALID'.

Example

Input	Output
7 2 AWFUL BAD CRUMMY LOUSY POOR SAD TERRIBLE SO5OD TROUBADOUR	VALID INVALID

Problem J. Mathemagicians

Source file name:	mathemagicians.c, mathemagicians.cpp, mathemagicians.java, mathemagicians.py
Input:	`Standard`
Output:	`Standard`

There are n mathemagicians standing in a circle. Each mathemagician wears either a blue hat or a red hat. A mathemagician can cast a color change charm which changes the color of their hat to the same color as the hat of the mathemagician directly to the left or to the right of them (the caster of the spell may choose which one of them). Note that mathemagicians are polite people and do not like interrupting each other, so only one mathemagician at a time may perform mathemagic.

The mathemagicians are not happy with their current hat configuration, so they would like to use the color change charm repeatedly to enter another hat configuration. Time isn't an issue because they can conjure cookies to eat.

Input

The first line contains an integer n ($3 \leq n \leq 10^5$), the number of mathemagicians. The next contains a string of length n. If the ith mathemagician wears a blue hat in the beginning, the ith character of the string is '`B`', otherwise the ith character is '`R`'. Finally, the third line contains a string of length n. If the ith mathemagician would like to wear a blue hat in the end, the ith character of the string is '`B`', otherwise the ith character is '`R`'.

It is guaranteed that not every mathemagician is happy with their hat color in the beginning.

Output

Output "`yes`" if it is possible for the mathemagicians to achieve the desired hat configuration after a finite number of color change charms, otherwise output "`no`".

Example

Input	Output
5 BRBBR RBBRR	yes
6 RBRBRB BRBRBR	no

Problem K. Correcting Keats

Source file name:	keats.c, keats.cpp, keats.java, keats.py
Input:	`Standard`
Output:	`Standard`

Charles complains, "Keats makes so many spelling errors; it's impossible to fix them all!"

Ada responds, "We have to get the manuscript into the Engine. Maybe we can program the Engine to check each word against a list of English words, and if it's not found in that list, see what sort of small errors might have been made."

Charles asks, "But what is a small error?"

With some discussion they agree on the following list of small errors:

- Adding a letter anywhere in the string.
- Removing a letter from anywhere in the string.
- Changing any letter in the string to any other letter.

Given a specific alphabet and a particular string, find all other strings in that alphabet that can be created by making one of the mistakes in the above list, and list them in alphabetical order.

Note that the input string must not be in the list, and the list must not contain duplicates.

Input

The first line of the input is a sequence of lowercase alphabetic characters, in alphabetical order, with no spaces between them. This is the alphabet to use. The second line contains the input string, which will consist only of letters from the given alphabet, and have length at least 2 and at most 100.

Output

List, in alphabetical order, all strings that can result from making one error in the given word.

Example

Input	Output
eg egg	eeg eegg eg ege egeg egge eggg gegg gg ggg

Problem L. Programming Team's Will

Source file name:	will.c, will.cpp, will.java, will.py
Input:	`Standard`
Output:	`Standard`

The UCF Programming Team has a not-so-well-known tradition of creating "Last Wills" for the event that one of the members leaves the team during the year unannounced. If that happens, something must be done with their lollipop collection in the Programming Team Lab! The Programming Team lets its members specify to which other UCF students they want to give their lollipops. Each member writes up a list of UCF students and, for each student, what fraction of the lollipop collection should be given to them. Note that all team members are UCF students as well so a team member could have another member in their will.

Normally this is a perfect solution; someone drops out of UCF to switch to some lesser college, and their lollipops are spread among their specified recipients. However, there has been a horrible tragedy. Some of the team was enrolled in Dr. Meade's CS-I class, and just took the first exam. Everyone who was enrolled has failed out of the class, and is kicked off the team by Dr. Meade! But what should happen to their lollipop collections now? The issue is that some team members had each other in their wills, so we cannot simply resolve them one at a time.

To handle this, the team agreed upon a fair strategy to distribute the lollipops. The wills would all be repeatedly applied (including giving lollipops to departing team members) until the total lollipops of departing members converged to zero. In the event that infinite applications of everyone's wills would still leave some lollipops with departing members, those lollipops would be thrown away. However, this is potentially a very slow and infinite process, so they need your help to figure out where the lollipops end up. Note that a fractional part of a lollipop can be given, if needed, by crushing up the lollipop.

The Problem: Given a list of departing team members' wills and their lollipop counts, output how many lollipops each student at UCF will end up with.

Input

The first input line contains two space separated positive integers: N $(2 \leq N \leq 500)$ representing the number of programming team members in Dr. Meade's class, and M $(N < M \leq 50,000)$ representing the total number of UCF students. The programming team members in Dr. Meade's class have ID's 1 through N, and the other students have ID's $N + 1$ through M. Then follow N wills, each described as follows: Each will starts with a line containing two integers: L $(1 \leq L \leq 1000)$ representing the number of lollipops, and K $(1 \leq K < M)$ the number of entries in this will. The following K lines each contain an integer X specifying the id of the person in the will, and a floating-point number P $(P \geq 10^{-6})$ specifying what portion (fraction) of the lollipop collection goes to person X in this will. Assume that:

- A single person's will contains only unique people, i.e., there won't be two entries for the same person in a will.
- The will for a person will not contain themselves.
- The fractions in a team member's will add up to 1.
- The total number of entries in all wills will not exceed 1,000,000.
- The input values for P will be given with no more than 6 digits after the decimal point.

Output

Output M lines, containing the number of lollipops that each student ends up with (in order by ID).

Note that the first N lines should be 0, since all lollipops from those students will either be given away or thrown away. Your answer will be judged to a precision of 10^{-5} .

Example

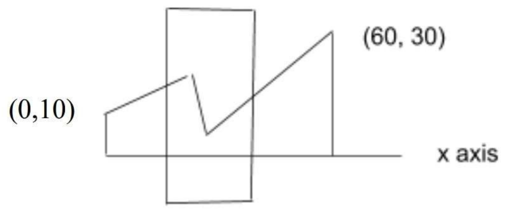
Input	Output

2 5
100 2
2 0.9
3 0.1
100 2
1 0.2
4 0.8
	0.0
	0.0
	14.63414634
	185.36585366
	0.0

Problem M. Mountain View

Source file name:	view.c, view.cpp, view.java, view.py
Input:	`Standard`
Output:	`Standard`

You have spent the summer in Mountain View and want to share with your friends the beautiful view. The mountains are far away enough that we can model them as lined up on a straight line on the positive x-axis. When you take a picture of the mountains, your camera captures a fixed width, W, of the mountain view, capturing the mountain outline in the range $[x, x + W]$. If we assign the elevation of a point to be its y-value, then we can describe the mountain outline as a series of ordered pairs, from left to right, where the mountain line between ordered pairs is the straight line between the two points. Consider the mountain outline defined by the points $(0, 10)$, $(20, 20)$, $(30, 5)$, and $(60, 30)$:



PIcture of width W = 20

If your camera width was $W = 20$, then one possible picture you could take is the one shown above in between the range of $x = 15$ and $x = 35$. Since you want to show your friends a mountainous view, you would like to show them a picture where the average height of the mountain is maximal of all possible pictures you could take with a fixed width of W. For the example above, the best choice would be the picture from $x = 40$ to $x = 60$. For this picture, the average elevation of the mountain line is $y = 65/3$.

The Problem: Given the description of a mountain line and a fixed width W for your picture, determine, of all possible pictures you could take, the maximum average elevation.

Input

The first line of input contains two positive integers: n $(2 \leq n \leq 10^5)$, the number of points describing the mountain line, and W $(1 \leq W \leq 10^9)$, the width of the picture your camera takes. The i^{th} $(1 \leq i \leq n)$ line of the following input lines contains two non-negative integers, x_i $(x_i i \leq 10^9)$ and y_i $(y_i \leq 10^4)$, the x and y coordinates, respectively, of the i^{th} point describing the skyline. It is guaranteed for all $i < n$ that $x_i < x_{i+1}$. Assume that outside of the mountain landscape given in the input, the elevation is always 0.

Output

On a line by itself, output the maximum average height of all possible pictures you could take. Print 9 digits after the decimal point. Answers will be judged correct if they are within an absolute tolerance of 10^{-6}.

Example

Input	Output

4 20
0 10
20 20
30 5
60 30
	21.666666667

3 1
10 50
90 50
1000 49
	50.000000000