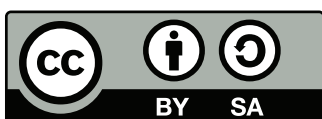


Groningen Algorithm Programming Contest 2025



Problems

- A Array Annihilation
- B Bakfiets
- C Characterithmetic
- D Delicious Trees
- E Equation Extrapolation
- F Frog and Princess
- G Gambler's Dilemma
- H Hopelessly Hungover
- I Interesting Mountains
- J Jumbled Keys
- K Kite Construction



Copyright © 2025 by The Freshmen Programming Contests 2025 jury (AAPJE in Amsterdam, FPC in Delft, FYPC in Eindhoven, GAPC in Groningen, and in Mons). This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.
<https://creativecommons.org/licenses/by-sa/4.0/>

A **Array Annihilation**

Time limit: 2s

Already another algorithms assignment about arrays – Amy has absolute aversion to them all! To avenge the amountful assignments, she activates an Array Annihilator Pledging Justice Executable, so all annoying arrays are abolished.

As the arrays are astronomically large, she agrees to only use one type of operation repeatedly. For each operation, she chooses two or more consecutive values in the array, and subtracts 1 from each of them. Given an array, is it possible to make all values equal to 0 in some finite number of these operations?

As an example, consider the second sample input. The array can be fully annihilated in four operations: first, decrement the first two values (yielding [1, 3, 3]), then decrement the last two values twice (yielding [1, 1, 1]), and finally decrement the entire array once (yielding [0, 0, 0]).



Amy, actively annihilating an array.
Image generated using DALL·E

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 5 \cdot 10^5$), the length of the array.
- One line with n integers a ($1 \leq a \leq 10^9$), the values in the array.

Output

If it is possible to make all values of the array equal to 0 in some finite number of operations, output “possible”. If this is not possible, output “impossible”.

Sample Input 1	Sample Output 1
2 1 1	possible
Sample Input 2	Sample Output 2
3 2 4 3	possible
Sample Input 3	Sample Output 3
1 5	impossible

Sample Input 4

```
4
2 1 1 2
```

Sample Output 4

```
impossible
```

Sample Input 5

```
9
3 4 5 5 1 2 3 2 1
```

Sample Output 5

```
possible
```

B Bakfiets

Time limit: 1s

Tim often needs to transport Fresh Yummy Pomegranate Cocktails in his bakfiets for events of his student association. For this, he has put a rectangular grid of $a \times b$ in his bakfiets so each bottle fits neatly in a grid cell. The cocktails can be bought in rectangular grid packaging of $w \times h$, strapped together with some plastic tape. However, these are not necessarily the same dimensions as the grid in his bakfiets. Since Tim was smart enough to make sure that his grid could fit all the bottles, he can remove some of them from the plastic packaging, and put them separately in his bakfiets. What is the minimum number of bottles he needs to remove from the packaging to fit everything in his bakfiets?



Tim and his bakfiets, called Timmy.
© W.I.S.V. ‘Christiaan Huygens’,
photo by Maarten Weyns,
used with permission

As an example, consider the first sample case, visualized in Figure B.1. Tim will need to remove three bottles from the plastic packaging and place these in the remaining space in the bakfiets, in order to make everything fit.

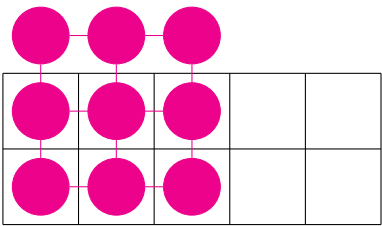


Figure B.1: Visualization of the first sample case. The pink circles represent the bottles in plastic packaging with dimensions $w \times h = 3 \times 3$ and the black squares represent the bakfiets with dimensions $a \times b = 5 \times 2$.

Input

The input consists of:

- One line with four integers w , h , a , and b ($1 \leq w, h, a, b \leq 10^9$, $w \cdot h \leq a \cdot b$). The dimensions of the plastic packaging are $w \times h$ bottles, and the dimensions of the bakfiets are $a \times b$ bottles.

Output

Output the minimum number of bottles Tim needs to remove from the packaging so that all bottles fit in his bakfiets.

Sample Input 1	Sample Output 1
3 3 5 2	3

Sample Input 2

1 4 3 2

Sample Output 2

1

Input 3

1000000000 1000 1000000 1000000

Output 3

999000000000

C Characterithmetic

Time limit: 3s

While solving one problem after the other in your favourite programming contest (hopefully, the one you are at right now!), you are suddenly faced with an Alternative Arithmetic Problem for the Jury's Entertainment. In this problem, you are not calculating with numbers, but with strings.

For some arbitrary string t and integer k , we define $t \odot k$ (t times k) to be the string t concatenated k times. If t can be written as $t' \odot k$ for some other string t' , we say that k is a *divisor* of t . For example, the divisors of the string "abababab" are 1, 2, and 4, because this string can be written as either "abababab" \odot 1, or "abab" \odot 2, or "ab" \odot 4. Finally, we say that a string is *indivisible* if its only divisor is 1: for example, the string "abc" is indivisible.



Trippi Troppi trying to change a string's largest divisor.
Image generated using Canva AI

You are given a string s of length n , that contains only the characters 'a', 'b', and 'c'. For each divisor d of n , you need to change some of the characters (to either 'a', 'b', or 'c'), so that the largest *divisor* of s becomes d . Find the minimum number of characters that need to be changed in order to make this true.

As an example, consider the first sample input. There are three answers, one for each of the divisors of 4:

- To make the largest divisor of "acac" be 1, we need to make sure that 2 is not a divisor. We can change any character to make the string *indivisible*, e.g. "bcac".
- The largest divisor of "acac" is already 2, so no characters need to change and the answer is 0.
- To make the largest divisor 4, we need to make all characters equal. To do this, 2 characters need to be changed, yielding either "aaaa" or "cccc".

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 10^5$), the length of your string.
- One line with a string s of length n , consisting only of characters 'a', 'b', and 'c'.

Output

For each divisor d of n , in increasing order, output the minimum number of characters that you need to change to make the largest divisor of s equal to d .

Sample Input 1

4 acac	1 0 2
-----------	-------

Sample Output 1**Sample Input 2**

6 abccba	0 2 4 4
-------------	---------

Sample Output 2

D Delicious Trees

Time limit: 3s

There is a Group of Adventurous, Playful Coatis living nearby a magnificent fruit tree. It provides the band with delicious fruits throughout the year. The coatis have maximized the tree's potential by keeping it in the shape of an AVL tree.¹

The group has grown a lot over the years, and this single tree is no longer enough to satisfy the hunger of the many coati kits. One particularly smart coati, named Stefan, comes up with the idea to apply the concept of parallelization to this tree: rather than having one big tree, they should cut it into smaller trees, each of which can then grow new fruit to feed the hungry band. Obviously, after the big tree has been cut up, each of the smaller trees should be an AVL tree. Find any way to cut the AVL tree into some predetermined number of smaller AVL trees, or say this is impossible.

As an example, consider the first sample input, a fully balanced binary tree with height 3. Cutting the edges to the parents of vertices 2 and 3 produces three AVL trees: one tree containing only vertex 1, and two fully balanced trees with height 2. Note that it is allowed that a tree is temporarily unbalanced in between the first and the last cut.



The coatis climbing in their AVL tree, deciding how to cut it.
CC BY-NC-ND 2.0 by Cloutail
the Snow Leopard on Flickr

Input

The input consists of:

- One line with two integers n and k ($2 \leq n, k \leq 10^5$), the number of vertices in the big AVL tree and the number of smaller trees they want to end up with.
- n lines, the i th of which contains two integers l and r ($0 \leq l, r \leq n$), the indices of the left and right child of the i th vertex, or 0 if that child is absent.

It is guaranteed that the input is a valid AVL tree rooted at vertex 1.

Output

If it is impossible to end up with k smaller AVL trees, output “impossible”. Else, output $k - 1$ numbers between 2 and n (inclusive), indicating that the edges to the parents of these $k - 1$ vertices should be cut, to get k AVL trees.

If there are multiple valid solutions, you may output any one of them.

¹An AVL tree is a binary tree with the constraint that at each vertex, the difference between the two depths of its children's (possibly absent) subtrees is at most 1. Note that the depth of an absent subtree is equal to 0.

Sample Input 1

```
7 3
2 3
5 4
6 7
0 0
0 0
0 0
0 0
```

Sample Output 1

```
2 3
```

Sample Input 2

```
3 42
2 3
0 0
0 0
```

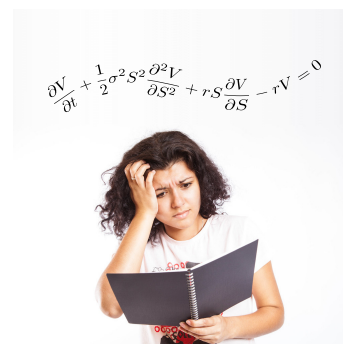
Sample Output 2

```
impossible
```

E Equation Extrapolation

Time limit: 1s

Following recent events, you got hooked on predicting the stock market and entered *The Quant Cup*, one of the most prestigious trading competitions in the world, where participants must predict the future of a simulated market. The challenge this year is more daunting than ever before, but after experimenting with the data provided by the organizers, you have stumbled upon a groundbreaking flaw. The data reveals that the asset prices in the simulation do not follow any random pattern. Instead, they seem to follow a hidden polynomial function $P(x)$, where x represents a variable that influences the stock price. Figuring out this polynomial's coefficients would give you an unfair advantage in the competition, and you would definitely win. However, computing $P(x)$ is a tedious and time consuming task, and the competition runs for just long enough so that you can make 9 such computations. Can you decode the hidden polynomial and take the lead in the competition?



Pondering the complicated equations governing the stock market.
CC BY-SA 2.0 by CollegeDegrees360 on Flickr, modified

More formally, there exists a hidden polynomial $P(x) = a_0 + a_1x + a_2x^2 + \dots + a_dx^d$ of degree d ($0 \leq d \leq 10$) with integer coefficients a_i ($0 \leq a_i \leq 9$ for each i). Find the original polynomial and print its $d + 1$ coefficients. To do this, you may query at most 9 integer values x ($-10^6 \leq x \leq 10^6$), and for each query, you will receive the value of $P(x)$.

Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads from the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

Your program should make at most 9 queries to find the original polynomial. The polynomial's degree is not provided up front, it must be inferred from the queries. Each query is made by printing one line of the form “? x ” with some integer x ($-10^6 \leq x \leq 10^6$). The interactor will respond with the integer value of $P(x)$.

When you have determined all of the coefficients of the original polynomial, print one line of the form “! a_0 a_1 ... a_d ”, where a_i is the coefficient of the x^i term in the original polynomial, after which the interaction will stop. Printing the answer does not count as a query.

The interactor is not adaptive: the coefficients are fixed up front, and do not depend on your queries.

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

Using more than 9 queries will result in a wrong answer.

Read	Sample Interaction 1	Write
	<div>? 4</div>	
<div>313</div>		
	<div>? 7</div>	
<div>1534</div>		
	<div>! 1 2 3 4</div>	
Read	Sample Interaction 2	Write
	<div>? 2</div>	
<div>0</div>		
	<div>! 0</div>	
Read	Sample Interaction 3	Write
	<div>? 467</div>	
<div>1094649</div>		
	<div>! 1 9 5</div>	

F Frog and Princess

Time limit: 2s

In the distant land of Glimmerbrook, there once lived a noble prince, brave of heart. Alas, his fate took a cruel turn when he crossed paths with an ancient and vengeful wizard. With a flick of his staff, the wizard cast a dreadful curse – transforming the prince into a frog, slimy and green. The curse was no ordinary spell, but part of an age-old magical trial known as the Geometric Arcane Puzzle of Coordinates – a test designed by the old wizards to trap the bold and free-spirited. Now, bound by enchantment, the frog-prince must reach the princess – for only her touch can break the curse and restore his true form.

The magic has twisted his very legs: he may leap no more than n times. The length of each jump (measured in Euclidean distance) is strictly dictated by the ancient spell: on the i th jump, the frog should jump a distance of a_i , no more, no less. If he can land at the princess's side before his jumps run out, the curse will shatter instantly (even if some jumps remain) and he shall be free. But should he fall short, he will remain a frog forevermore.



Public Domain by Paul Friedrich Meyerheim on Wikimedia Commons

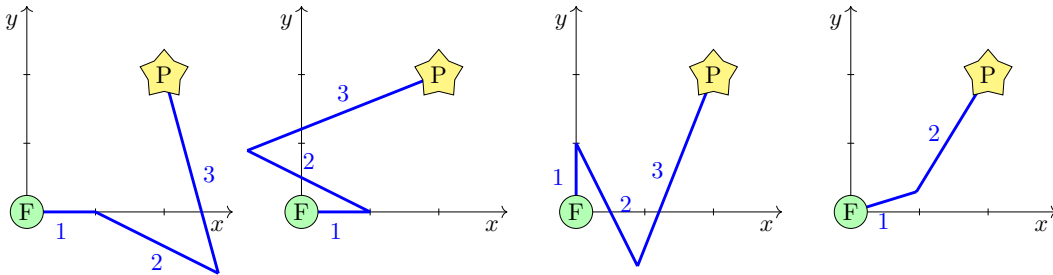


Figure F.1: Some ways that the frog can jump to the princess in the first sample.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 4 \cdot 10^5$), the number of jumps that the frog can make.
- One line with four integers x_f, y_f, x_p , and y_p ($-10^9 \leq x_f, y_f, x_p, y_p \leq 10^9$), the coordinates (x_f, y_f) of the starting position of the frog, and the coordinates (x_p, y_p) of the princess.
- One line with n integers a_1, \dots, a_n ($1 \leq a_i \leq 10^9$ for each i , and $\sum_{i=1}^n a_i \leq 2 \cdot 10^9$), where a_i is the length of the i th jump of the frog.

It is guaranteed that the coordinates of the frog and those of the princess are not the same.

Output

Output “yes” if the frog can jump to the exact position of the princess, and “no” otherwise.

Sample Input 1

3 0 0 2 2 1 2 3	yes
-----------------------	-----

Sample Output 1**Sample Input 2**

2 0 0 -2 -2 1 1	no
-----------------------	----

Sample Output 2**Sample Input 3**

3 0 0 2 2 6 1 2	no
-----------------------	----

Sample Output 3**Sample Input 4**

1 0 1 1000000000 0 1000000000	no
-------------------------------------	----

Sample Output 4

G Gambler's Dilemma

Time limit: 1s

You are new to poker and want to know when to fold before the flop. The flop refers to the turning of the first three cards on the table, which happens after you have received two cards in your hand. After receiving your two cards, you need to decide whether you want to keep them and place a bet or to fold (which means to step out of the game without placing a bid).

There are complicated tables for this, which also depend on how many people are sitting between you and the dealer, but you have decided on the following heuristic.² You keep your two initial cards and place a bet if they have one of the following properties:

- They are a pair (two cards of the same rank), but not a pair of 2s or a pair of 3s.
- The two cards are of consecutive ranks, and each is 9 or higher.
- The two cards are of the same suit, and at least one card is a Queen or higher.
- Each card is a Jack or higher.

If your two initial cards do not have any of these properties, you fold before the flop.

Note that the Ace ranks higher than the King, and not lower than the 2.

Input

The input consists of:

- One line with two different cards, each of which consists of one character $r \in \{2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A\}$ (denoting the rank: a number, Ten, Jack, Queen, King, or Ace) and one character $s \in \{C, D, H, S\}$ (denoting the suit: clubs, diamonds, hearts, or spades).

Output

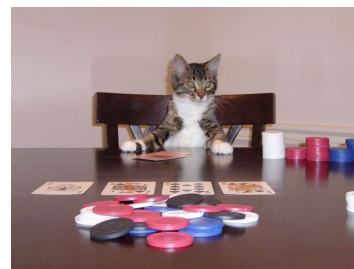
Output “yes” if you keep your two initial cards, and “no” if you fold before the flop.

Sample Input 1

TD TS

Sample Output 1

yes



A beginning poker player that accidentally went all-in. CC BY 2.0 by cyOFdevelin fame on Flickr

²We deny any responsibility for using this heuristic in practice.

Sample Input 2

8D 9D

Sample Output 2

no

Sample Input 3

5H 3S

Sample Output 3

no

H Hopelessly Hungover

Time limit: 1s

First-year student Bob needs to learn n facts for an online exam. Currently, he knows none. The exam can be taken at any time, even during weekends, but due to the surveillance software, it is impossible to cheat. The grading is quite harsh and requires that he knows all the n facts to pass.

On weekdays, Bob studies diligently and can learn at most k facts per day. On weekends, there are the Freshmen Party Committee events, meticulously planned and impossible to miss, where he forgets m facts per day due to sheer excitement. Of course, when the number of facts Bob knows is less than m , he will forget all the facts during the party and wakes up the next day knowing none. Starting in the morning of some given day of the week, determine how many days it takes until he first knows all n facts, or determine that he never learns all n facts.

As an example, consider the first sample input, where Bob needs to learn 14 facts. He can learn 1 fact every weekday and forgets 1 fact per day in the weekend. He starts off knowing 0 facts on a Monday morning. Bob learns 1 fact during the day. He does so on all the 5 weekdays, so on Saturday morning he knows 5 facts. Unfortunately, he forgets a fact on both Saturday and Sunday, so he starts the second week on Monday morning knowing 3 facts. This keeps going in the span of 4 weeks. On the Friday of the last week, he knows all the 14 facts so he can start his online exam in the evening. This turns out to be exactly 26 days, accounting for the starting Monday and final Friday as well.



Beer and studying do not go great together.
Pexels License by seymasungr on Pexels

Input

The input consists of:

- One line with four integers n , k , m , and d ($1 \leq n, k, m \leq 1000$, $1 \leq d \leq 7$), where: n is the total number of facts that Bob must learn, k is the number of facts learned on a weekday, m is the number of facts forgotten on a weekend day, and d represents the current day of the week (where 1 corresponds to Monday and 7 corresponds to Sunday).

Output

Output the number of days (including the current day) that elapse before Bob first knows all n facts. Note that the last day that Bob still needs to study should be included, although he will know all the facts on the same day in the evening and can start his exam immediately. If Bob can never learn all facts, output “impossible”.

Sample Input 1

14 1 1 1

Sample Output 1

26

Sample Input 2

15 1 10 4

Sample Output 2

impossible

Sample Input 3

10 2 20 2

Sample Output 3

11

I Interesting Mountains

Time limit: 2s

On your holiday trip to the mountains, you are amazed by the high quality of your Flashy Yellow Panorama Camera. After shooting a couple of nice photos during your hike, you decide to send the nicest one to your best friend. Sure enough, they immediately reply that they see all kinds of fancy patterns – but they are not talking about the beautiful snow-capped mountains: they have a more abstract view of the photo. . .



The Vestrahorn in Iceland is an interesting formation.
Image by tawatchai07 on Freepik, modified

Since the heights of the n mountains in the panorama photo are unique, you can view these heights as a permutation of all numbers between 1 and n (inclusive). Your friend thinks that a formation of three (not necessarily consecutive) mountains is *interesting*, when the first mountain is higher than the third mountain, and the third mountain is higher than the second mountain. In other words, three mountains with indices i , j , and k ($1 \leq i < j < k \leq n$) and respective heights h_i , h_j , and h_k are interesting, if and only if $h_i > h_k > h_j$. How many interesting formations can you find in a given panorama photo?

As an example, consider the first sample input: only the tuples with indices (1,3,5) and (2,3,5) are interesting formations.

Input

The input consists of:

- One line with an integer n ($3 \leq n \leq 3 \cdot 10^5$), the number of mountains in the photo.
- One line with n integers h_1, \dots, h_n ($1 \leq h_i \leq n$ for each i), the heights of the mountains in the photo. It is guaranteed all values of h_i are unique.

Output

Output the number of interesting formations in the panorama photo.

Sample Input 1	Sample Output 1
5 3 4 1 5 2	2

Sample Input 2	Sample Output 2
3 3 1 2	1

Sample Input 3

3	0
2 1 3	

Sample Output 3**Sample Input 4**

11	69
5 9 10 11 1 2 3 4 6 7 8	

Sample Output 4

J Jumbled Keys

Time limit: 2s

Georgi is a freshman, who just started his Computer Science studies in The Netherlands. Being very ambitious, he already looked at last year's exam for the Introduction to Programming course, and was concerned by how little time he had to do it. To help him complete his exam on time, Georgi has devised a brilliant new way to type – the Finger Placement Configuration (FPC). The FPC is just a new keyboard layout, where pressing any letter on his QWERTY keyboard results in a unique English letter being input. Every lowercase English letter can be typed and every key only inputs one letter.



Image by u/ROD_OF_AGES on
r/MechanicalKeyboards

After a rigorous day of studying and practicing with the new iteration of the keyboard layout, Georgi decides to log into one of his favourite games: Warstone – Heroes of Hearthcraft. However, he has forgotten his password. He had written his password into his password manager software, but made a mistake when doing so: he had typed the password thinking that the keyboard was in QWERTY mode, but the FPC layout was still enabled! The password can probably be recovered knowing the configuration of the keys in the FPC layout, but, being exhausted, he has completely forgotten that too. The only tool he can use is his training sheet. Each entry in the training sheet consists of a sequence of keys to press on the FPC layout, followed by the resulting word. On the verge of giving up, he desperately asks you for help in figuring out what his password was.

Input

The input consists of:

- One line with a single integer n ($1 \leq n \leq 1000$), the number of training words.
- n lines, each with two strings of equal length: the first is the sequence of keys that need to be pressed to enter a word using the FPC layout, and the second is the resulting word.
- One word, the sequence of keys that need to be pressed to obtain the password using the FPC layout.

Each word in the input has a length between 1 and 1000 characters and only consists of English lowercase characters (a–z).

Output

Output Georgi's password.

It is guaranteed that it is possible to uniquely reconstruct the original password based on the data from the training sheet.

Sample Input 1

```
2
abc acb
cbde bced
abcde
```

Sample Output 1

```
acbed
```

Sample Input 2

```
1
abcdefghijkl amlvsioked
eabgcidehf
```

Sample Output 2

```
samolevski
```

Input 3

```
1
abcdefghijklmnopqrstuvwxy abcdefghijklmnopqrstuvwxx
abcdefghijklmnopqrstuvwxyz
```

Output 3

```
abcdefghijklmnopqrstuvwxyzy
```

K Kite Construction

Time limit: 4s

Summer is coming up, and you and your friends love to fly kites at the beach. You plan to make n new kites this year, and paint them in some range of colours. For this, you already have a practically unlimited amount of canvas and paint, but the limiting factor is where you can put the painted kites while they dry.

To hang out the kites for drying, you found a square field with side length ℓ . On each of the four sides of the field, there are n poles. The corners of the field are at coordinates $(0, 0)$, $(\ell, 0)$, $(0, \ell)$, and (ℓ, ℓ) , and you know the coordinates of the poles in this coordinate system. There are no poles exactly on a corner of the square, and no poles inside the square.



Neatly folded canvases and paint, waiting to be transformed into colourful kites.
Image generated using DALL · E 3

Since you are planning to make as many kites as there are poles on each side of the field, and each kite has four corners, you want to attach each of the corners of each kite to a pole from each of the four sides. Additionally, each pole should be attached to exactly one corner of a kite. Of course, larger kites are more fun, so you want to maximize the total area of the kites.

Although you vaguely remember from geometry class that a kite has some special properties, you decide to use your artistic freedom and just allow any (convex) quadrilateral as the shape of your kites.

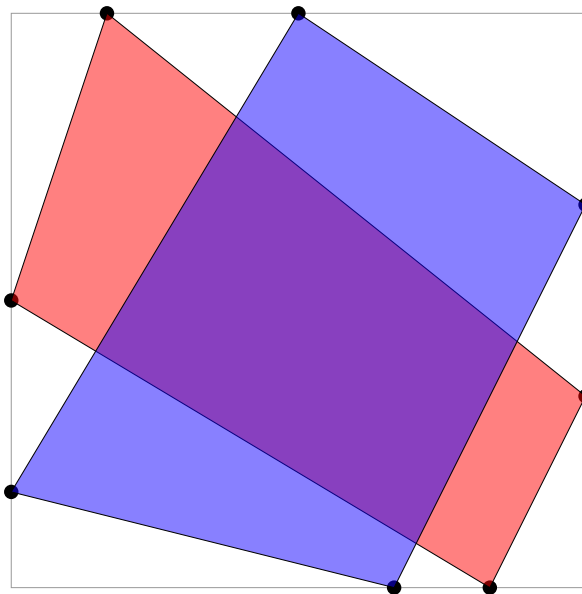


Figure K.1: Visualization of the second sample input, where the grey square indicates the 6×6 square field, the black dots indicate the poles, and the coloured rectangles indicate a possible placement of the kites. Note that this is *not* the configuration that maximizes the total area of the kites (the red kite has an area of 16, the blue kite has an area of 19.5, for a total area of 35.5).

Input

The input consists of:

- One line with two integers n and ℓ ($1 \leq n \leq 10^5$, $1 \leq \ell \leq 10^6$), the number of kites you want to make and the side length of the square field.
- $4n$ lines, each with two integers x and y ($0 \leq x, y \leq \ell$), the coordinates of the poles on the sides of the square.

It is guaranteed that all poles lie on the boundary of the square and n poles lie on each side of the square, and no poles lie on a corner of the square. It is also guaranteed that no two poles are at the same location.

Output

Output the maximum total area of the n kites.

Your answer should have an absolute or relative error of at most 10^{-9} .

Sample Input 1

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

Sample Output 1

```
8
```

Sample Input 2

```
2 6
0 3
0 1
6 2
4 0
3 6
6 4
1 6
5 0
```

Sample Output 2

```
41.5
```