icpc | europe contests

The 2023 ICPC Northwestern Europe Regional Contest

Official Problem Set

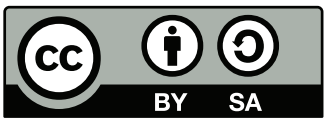
Northwestern Europe Regional Contest 2023

NWERC 2023

November 26, 2023

TU Delft Delft University of Technology

Problems



Problem A
Arranging Adapters
Time limit: 4 seconds

It is the day before the NWERC and you and your team are on the train towards Delft. The journey is long and boring but you came up with a good idea: "Let's do some *train*ing".

– silence –

You take your laptop out and try to plug it in when you notice that the only socket is already in use. Your friends smirk and reply: "No socket for you, no *train*ing for us". Their smirks quickly fade as you pull out a power strip, unplug the charger from the socket, and plug it back into the power strip. Now, there is enough space for your charger as well.

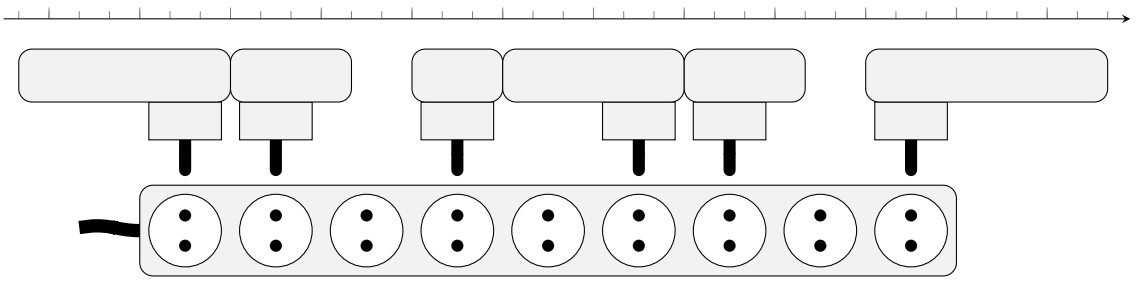


Figure A.1: Illustration of Sample Input 2. The first six chargers can be plugged in as shown. Note that this is not the only possible solution. However, it can be shown that it is impossible to plug in all seven chargers.

However, as soon as more sockets are available, your friends suddenly take out more devices that need to be charged. You realize that you will not get them to train like this, so you decide to trick them into solving a problem instead.

Your power strip comprises a row of s sockets, and each socket is $3\,\mathrm{cm}$ in diameter. Furthermore, as you examine the chargers, you notice that they all have integer lengths. The plug of each charger is always on one of the two ends, and each charger can only be used in two orientations. Chargers cannot overlap, but can touch, and can extend beyond the end of the power strip as long as they are plugged in to a socket. Now you challenge them to charge as many devices as possible. This is visualized in Figure A.1. Hoping that this allows them to avoid the *train*ing, your friends agree to write a program to solve this.

Input

The input consists of:

- One line with two integers n and s ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq s \leq 10^9$), the number of chargers you have and the number of sockets on the power strip.
- One line with n integers w ($3 \leq w \leq 10^9$), the width of each charger in centimetres.

Note that you are allowed to rotate chargers by 180° before plugging them in.

Output

Output the maximum number of chargers you can plug into the power strip at the same time.

Sample Input 1	Sample Output 1
5 7 7 4 4 5 8	5

Sample Input 2	Sample Output 2
8 9 7 4 3 6 4 8 5 6	6

Problem B
Brickwork
Time limit: 7 seconds

Bob the Builder is tired of building tiny houses and paving narrow roads, and he strives for something bigger. The new job given to him by a very eccentric client is exactly what he needs: He is tasked with building a wall of a certain width that is infinitely high! His client assured him that he does not need to worry about the building material, and that an infinite supply of various kinds of bricks has already been ordered for him. Of course, building a stable wall takes very careful planning, especially if it is supposed to be infinitely high. In particular, a wall is only stable if no two gaps between bricks in consecutive rows end up directly above each other, as shown in Figure B.1. Bob knows from his long-time experience that if it is possible to build such a wall, then it can be done by alternating just two row configurations.

Figure B.1: On the left, we see an unstable wall using the brick types of Sample Input 1. On the right, we see a stable wall using the same brick types. Note that even though only two rows of the wall are shown, it is possible to build an infinitely high wall by repeating these two row configurations.

Bob is terribly excited about the new job and quickly goes to work. Given the types of bricks available, is it possible to build a stable wall of width exactly w and infinite height? If yes, how should Bob build it using only two alternating row configurations?

Input

The input consists of:

- One line with two integers n and w ($1 \leq n, w \leq 3 \cdot 10^5$), the number of brick types and the width of the wall.
- One line with n integers b ($1 \leq b \leq w$), the widths of the brick types.

Note that Bob has an infinite supply of all brick types.

Output

If it is possible to build a wall, then output "`possible`". Otherwise, output "`impossible`".

If a wall can be built, provide two row configurations that can be used in an alternating fashion. For both rows, first output the number of bricks needed for that row, followed by the lengths of the bricks in the order you want to use them. Your solution is considered valid if alternating the two rows infinitely would result in a stable wall.

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

Sample Input 1	Sample Output 1
4 12 3 2 7 2	possible 5 2 2 3 2 3 3 3 2 7

Sample Input 2	Sample Output 2
3 11 6 7 8	impossible

Problem C
Chair Dance
Time limit: 8 seconds

In a deterministic version of *Musical Chairs*[1], there are n chairs placed in a circle. The chairs are numbered from 1 to n in clockwise order. Initially, the ith player sits on the ith chair. During the game, the game master gives commands to all players at once.

A family playing Musical Chairs.
CC BY-SA 3.0 by Artaxerxes on Wikimedia Commons

The first type of command tells each player to move x chairs farther in clockwise order, so they must move from chair i to chair $i + x$.

The second type of command tells each player to move from chair i to chair $i \cdot x$. Both these calculations are done modulo n, where a remainder of 0 corresponds to chair n.

If two or more people want to move to the same chair, then the player needing to travel the least in clockwise direction to reach the chair gets to take the seat, and the other players trying to reach the same chair are out of the game. This is illustrated in Figure C.1, where the larger circles represent the chairs and their numbers are written on their inside. The smaller circles represent the players. The next command (`* 10`) tells player 10 (now on seat 11) and player 4 (now on seat 5) to move to chair 2. However, since player 10 needs to travel less, this player gets to take the seat. Note that the other 10 players will also move to some other chairs, but this is omitted from the figure for the sake of readability.

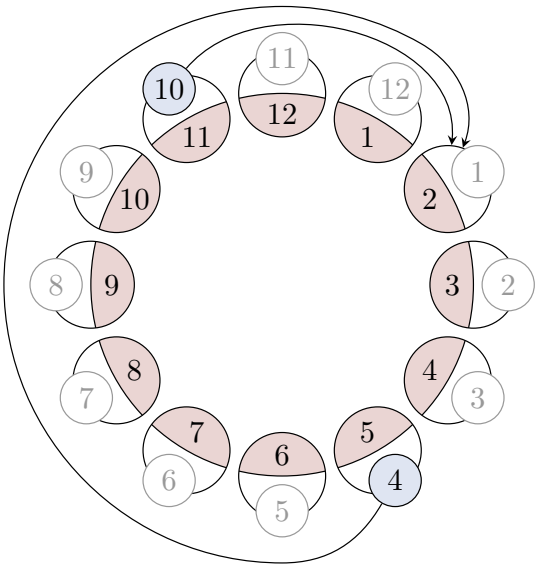


Figure C.1: Illustration of Sample Input 1 at the fourth command, where players 4 and 10 need to move to chair 2. Because player 10 needs to travel less in clockwise direction, this player gets to take the seat.

The jury wasted most of their free time designing this game and now need to go back to work. Fortunately, the game is deterministic, so you can play the game without the help of the jury.

[1]You do not need to know the original game, but you can try to play it after the contest is over.

Input

The input consists of:

- One line with two integers n and q ($2 \leq n, q \leq 5 \cdot 10^5$), the number of chairs and the number of commands.
- q lines, each containing one of three command types:
  - "`+` x": The player on chair i moves to chair $i + x$.
  - "`*` x": The player on chair i moves to chair $i \cdot x$.
  - "`?` x": Tell us the number of the player on chair x.

  All of the values x will satisfy $1 \leq x \leq n$.

Output

For each command of type '`?`', output the number of the player on the requested chair. If the chair is currently empty, output -1 instead.

Sample Input 1

```
12 10
? 12
+ 1
? 12
* 10
? 2
* 5
? 2
* 6
? 1
? 12
```

Sample Output 1

```
12
11
10
6
-1
11
```

Sample Input 2

```
32 11
* 6
? 8
* 6
+ 31
* 28
? 4
+ 1
* 2
+ 1
* 3
? 1
```

Sample Output 2

```
28
32
32
```

Problem D
Date Picker
Time limit: 1 second

A filled agenda.

The NWERC is coming up and your agenda is filling up with meetings. One of your teammates wants to plan a meeting, and asks for your input. However, instead of asking you for your exact agenda, you have to fill out two separate polls: one for indicating which days you are available, and one for the hours!

As a computer scientist, you plan your meetings only on whole hours and each meeting takes an integer number of hours. Therefore, your agenda can be modelled as a matrix of 7 rows (days), and 24 columns (hours). Each cell in this matrix is either '`.`' or '`x`', meaning that hour of that day you are either free or have a meeting, respectively.

You have to pick at least d days in the first poll and h hours in the second poll, and we assume the meeting will take place on any of your picked hour/day combinations with equal probability. What is the probability that you can attend the meeting if you fill in the polls optimally?

Input

The input consists of:

- 7 lines with 24 characters, each character being either '`.`' or '`x`', with '`.`' indicating the time slots you are available.
- One line with two integers d and h ($1 \leq d \leq 7$, $1 \leq h \leq 24$), the minimum number of days and hours you have to fill in.

Output

Output the probability that you are available at the chosen meeting time.

Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
xxxxxx..xx..xxxxxxxxxxxx xxxxxxxxxxxxx....xxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx xxxxxx..xx..xxxxxxxxxxxx xxxxxxxxxxxxx...x..xxxxx xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx 2 5	0.8

Sample Input 2	Sample Output 2
xxxxxxxxx.....x...xxxxxx xxxxxxxx..x...x...xxxxxx xxxxxxxx......x...x.xxxx xxxxxxxx...xxxxxxxxxxxx xxxxxxxx...xxxxxxxxxxxx xxxxxxxx...xxxxxxxx.xxxx ......xxxxxxxxxxxxxxxxx 3 8	0.958333333333333

Problem E
Exponentiation
Time limit: 3 seconds

Image by callmetak on Freepik

In her spare time, Zoe develops an online calculator. Unfortunately, the calculator was targeted by a denial-of-service attack last week. The attacker created a lot of integer variables, exponentiated them with each other, and tried to do a bunch of comparisons. The huge integers were too much for the server to handle, so it crashed. Before Zoe fixes the issue, she decides to actually perform the calculations that the attacker requested.

There are n integer variables $x_1, x_2, \ldots, x_n$. At the start, each variable is set to 2023. You have to perform m instructions of the following two types:

- *Operations*, of the form "`! i j`", where $i \neq j$. This means that x_i gets set to $x_i^{x_j}$.
- *Queries*, of the form "`? i j`", where $i \neq j$. This means that you should print '>' if x_i is greater than x_j, '=' if x_i is equal to x_j, and '<' if x_i is smaller than x_j.

Consider the first sample. After the 5 operations, the values of the variables are:

$$x_1 = \left(2023^{2023}\right)^{2023^{2023}}, \quad x_2 = \left(2023^{2023^{2023}}\right)^{2023}, \quad x_3 = 2023, \quad x_4 = 2023^{2023}.$$

Input

The input consists of:

- One line with two integers n and m ($2 \leq n \leq 1000$, $1 \leq m \leq 1000$), the number of variables and the number of instructions.
- m lines, each containing a character c (either '`!`' or '`?`') and two integers i and j ($1 \leq i, j \leq n$, $i \neq j$), describing the instructions.

Output

For every query in the input, output its answer.

Sample Input 1	Sample Output 1
4 8 ! 1 4 ! 2 1 ! 4 3 ! 1 4 ! 2 3 ? 3 4 ? 2 4 ? 2 1	< > =

Sample Input 2	Sample Output 2
4 9 ! 2 4 ! 1 2 ? 3 1 ? 1 2 ! 2 3 ? 1 2 ! 1 3 ! 3 2 ? 1 3	< > > <

Problem F
Fixing Fractions
Time limit: 5 seconds

Maths is hard.[citation needed] But it could be easier! And the internet™ has found some excellent ways to make it easier. Take a look at the following true equations:

Source: The Internet™.

$$\log(1) + \log(2) + \log(3) = \log(1 + 2 + 3)$$
$$\frac{16\cancel{3}}{\cancel{3}26} = \frac{1}{2}\,.$$

Following the patterns, we come to the conclusion that the following equation should also be true:

$$\frac{12\cancel{3}}{2\cancel{3}4} = \frac{1}{2}\,.$$

However, this is actually wrong in boring old standard maths. Therefore, we define a new kind of funky maths where it is allowed to cancel out digits on the left side of the equality sign. This surely will make everyone's life easier. Except yours, since you have to evaluate if two given fractions are equal in our new funky maths.

Input

The input consists of:

- One line with four integers a, b, c, and d ($1 \leq a, b, c, d < 10^{18}$), describing the two fractions $\frac{a}{b}$ and $\frac{c}{d}$.

Output

If there exist integers a' and b' obtained from a and b by cancelling out the same digits and with $\frac{a'}{b'} = \frac{c}{d}$ in standard mathematics, output "`possible`", followed by a' and b'. Otherwise, output "`impossible`".

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

Note that neither a' nor b' is allowed to contain leading zeroes after cancelling digits.

Sample Input 1	Sample Output 1
`163 326 1 2`	`possible` `1 2`

Sample Input 2	Sample Output 2
`871 1261 13 39`	`possible` `87 261`

Sample Input 3	Sample Output 3
`123 267 12339 23679`	`impossible`

This page is intentionally left blank.

Problem G
Galaxy Quest
Time limit: 5 seconds

You are travelling through the galaxy in your spaceship. There are n planets in the galaxy, numbered from 1 to n and modelled as points in 3-dimensional space.

You can travel between these planets along m space highways, where each highway connects two planets along the straight line between them. Your engine can accelerate (or decelerate) at $1\,\mathrm{m/s^2}$, while using fuel at a rate of 1 litre per second. There is no limit to how fast you can go, but you must always come to a complete standstill whenever you arrive at the planet at the end of a highway.

It is possible for a highway to pass through planets other than the ones it connects. However, as your spaceship is equipped with special hyperspace technology, it simply phases through these obstacles without any need of stopping. Another consequence of using this technology is that it is impossible to jump from one highway to another midway through: highways must always be travelled in full.

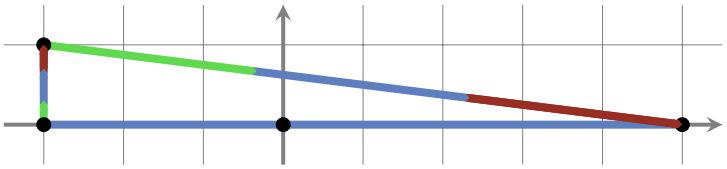


Figure G.1: Illustration of Sample Input 1, showing highways in blue, and a route from planet 1 to planet 3. The green start of a highway indicates acceleration, and the red end indicates deceleration.

You need to fly several missions, in which you start at your home planet (with number 1) and need to reach a given target planet within a given time limit. For each mission, determine whether it can be completed, and if so, find the least amount of fuel required to do so. As an example, Figure G.1 shows the optimal route for the second mission of the first sample.

Input

The input consists of:

- One line with three integers n, m, and q ($1 \leq n, m, q \leq 10^5$, $n \geq 2$), where n is the number of planets, m is the number of space highways, and q is the number of missions.
- n lines, each with three integers x_i, y_i, and z_i ($|x_i|, |y_i|, |z_i| \leq 10^3$, $1 \leq i \leq n$), the coordinates of planet i.
- m lines, each with two integers a and b ($1 \leq a, b \leq n$, $a \neq b$), describing a space highway that connects planets a and b. It can be traversed in either direction.
- q lines, each with two integers c and t ($2 \leq c \leq n$, $1 \leq t \leq 10^3$), the target planet and time limit for each mission.

The n planets are in distinct locations. Their coordinates are given in metres, and the time limits of the missions are given in seconds. No two highways connect the same pair of planets. For each mission, both the absolute and relative differences between the given time limit and the shortest possible completion time are at least 10^{-6}.

Output

For each mission, output the least amount of fuel in litres required to reach the target location within the time limit. If the target location cannot be reached within the time limit, output "`impossible`".

Your answers should have an absolute or relative error of at most 10^{-6}.

Sample Input 1

```
4 4 3
-30 0 0
0 0 0
50 0 0
-30 10 0
1 2
2 3
3 4
4 1
2 10
3 25
4 7
```

Sample Output 1

```
impossible
19.0538441903
4.0000000000
```

Sample Input 2

```
4 2 5
-3 0 2
7 -9 -3
4 4 -6
8 -1 8
1 2
2 3
2 1000
2 100
3 1000
3 100
4 1000
```

Sample Output 2

```
0.0287058122
0.2874671888
0.1120998619
1.1272896971
impossible
```

Problem H
Higher Arithmetic
Time limit: 4 seconds

Captchas are getting more and more elaborate. It started with doing simple calculations like $7 + 2$, and now, it has evolved into having to distinguish chihuahuas from double chocolate chip muffins.

To combat the rise of smarter bots, the Internet Captcha Production Company (ICPC) has outdone itself this time: given a distorted image containing many integers, find the maximum value that can be expressed using each of the given integers exactly once, using addition, multiplication, and arbitrary parentheses.

After unsuccessfully trying to solve such a captcha for an hour straight, Katrijn is terribly frustrated. She decides to write a program that outputs a valid arithmetic expression with maximal value.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^5$), the number of integers in the captcha.
- One line with n integers a ($1 \leq a \leq 10^6$), the integers in the captcha.

Output

Output a valid arithmetic expression with maximal value, where each integer from the input list is used exactly once. The usual order of operations applies. The output expression may use at most 10^6 characters and must not contain any spaces. Such an expression exists for any possible input.

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

Sample Input 1	Sample Output 1
4 1 2 3 4	3*((1+2)*4)

Sample Input 2	Sample Output 2
3 13 37 1	(1+13)*37

Sample Input 3	Sample Output 3
4 1 1 1 1	((1+1)*(1+1))

This page is intentionally left blank.

Problem I
Isolated Island
Time limit: 8 seconds

On a small island far far away, a handful of old men live isolated from the rest of the world. The entire island is divided into plots of land by fences, and each old man owns a single plot of land bounded by fences on all sides. (The region outside all fences is the ocean.) Each of the inhabitants needs fish to survive and the only place where they can fish is the ocean surrounding them. Since not every plot of land is connected to the ocean, some of the men might need to pass through the land of others before being able to fish. The men can cross a single fence at a time, but cannot go through fenceposts or locations where fences intersect.

Unfortunately, the old men are greedy. They demand one fish each time a person wants to enter their land. Since they do not want to lose too much fish to the others, every old man chooses a route that minimizes the number of fish he has to pay to get to the ocean.

Over the years, this has led to rivalry between the old men. Each man hates all other men who have to pay less than him to reach the ocean. Two men only *like* each other if they have to pay the same amount of fish to reach the ocean.

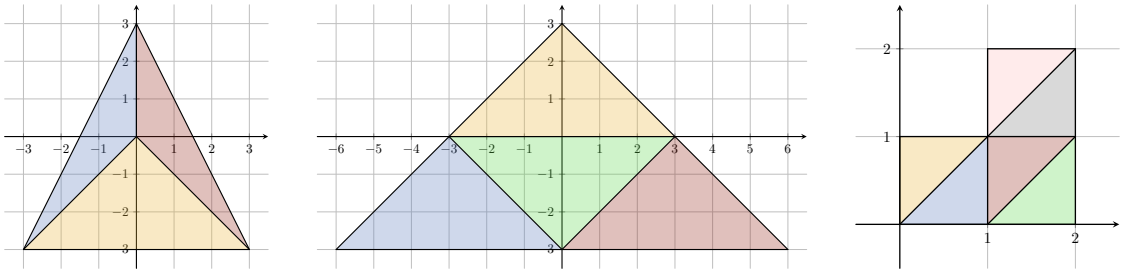


Figure I.1: Illustrations of the first three Sample Inputs. In Sample Input 1, every man has direct access to the sea, so they all like each other. In Sample Input 2, there does not exist a pair of neighbours who like each other, because the man living in the middle needs to pay one fish, whereas all of his neighbours do not have to pay any fish to reach the ocean. In Sample Input 3, there are six men, some of whom are friendly neighbours.

The natural question which now occurs is: are there some old men on this island who are neighbours (owning land on opposite sides of a single fence) and like each other? See Figure I.1 for two islands with opposite answers to this question.

Input

The input consists of:

- One line with an integer n ($3 \leq n \leq 1000$), the number of fences.
- n lines, each with four integers x_1, y_1, x_2, and y_2 ($|x_1|, |y_1|, |x_2|, |y_2| \leq 10^6$, $(x_1, y_1) \neq (x_2, y_2)$), indicating a straight fence between fenceposts at (x_1, y_1) and (x_2, y_2).

Note that fences may intersect internally, and that three or more fences may intersect in the same location.

It is guaranteed that any two fences intersect only in at most one point. Furthermore, after crossing a single fence, one always ends up in a different region. All regions together form a single island, where any region can be reached from any other region.

Output

If there exists a pair of neighbours who like each other, then output "`yes`". Otherwise, output "`no`".

Sample Input 1

```
6
-3 -3 0 3
-3 -3 0 0
-3 -3 3 -3
0 0 0 3
0 0 3 -3
0 3 3 -3
```

Sample Output 1

```
yes
```

Sample Input 2

```
6
-6 -3 0 3
0 3 6 -3
6 -3 -6 -3
-3 0 3 0
3 0 0 -3
0 -3 -3 0
```

Sample Output 2

```
no
```

Sample Input 3

```
8
0 1 2 1
2 2 0 0
1 2 1 0
1 0 2 1
0 0 2 0
1 2 2 2
0 1 0 0
2 2 2 0
```

Sample Output 3

```
yes
```

Sample Input 4

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

Sample Output 4

```
no
```

Problem J
Jogging Tour
Time limit: 8 seconds

You may know that in the 17th century, a group of Dutchmen founded a settlement called New Amsterdam on Manhattan Island that later went on to become New York City. Less well-known is the story of another group of Dutchmen that also moved over to America and founded a city called *New Delft*. Like its bigger counterpart, New Delft has been built on a grid made up of two sets of parallel streets that meet each other at a perpendicular angle.

Some stroopwafel bakeries have already been built in New Delft, but none of the streets have been constructed. Your task is to lay out the grid of streets. For this, you need to decide on an orientation for the grid so that there are two orthogonal directions for the two types of streets. Once the orientation is fixed, you may build arbitrary streets, as long as each of them has one of the two given directions, as shown in Figure J.1. Each street can be traversed in either direction.

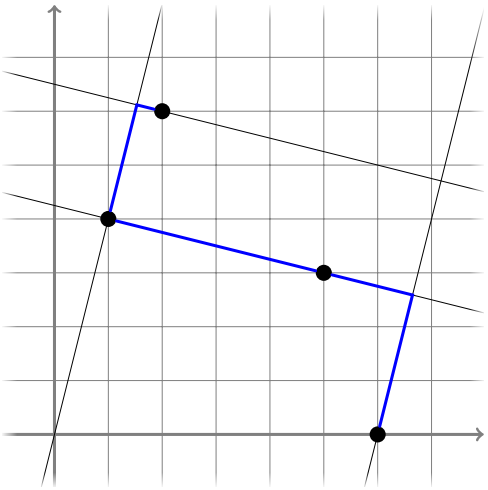


Figure J.1: Illustration of Sample Input 2 with a possible street layout that gives the shortest possible path that visits all bakeries in some order.

The street layout should be created in an optimal way for the annual *Stroopwafel Run*. This is an event in which a group of runners visits all the bakeries in some order of their choosing, and they may start and end their run at any point in the city. Your task is to come up with a grid layout that makes this shortest path as short as possible.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 12$), the number of stroopwafel bakeries in New Delft.
- n lines, each with two integers x and y ($0 \leq x, y \leq 10^6$), the coordinates of one of the bakeries.

The bakeries are at distinct coordinates, so for any $1 \leq i, j \leq n$ with $i \neq j$, it holds that $(x_i, y_i) \neq (x_j, y_j)$.

Output

Output the length of the shortest possible path that visits all bakeries in some order, assuming an optimal grid layout.

Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
3 0 1 1 2 3 0	4.24264068712

Sample Input 2	Sample Output 2
4 1 4 6 0 5 3 2 6	11.1566387517

Problem K
Klompendans
Time limit: 5 seconds

In traditional Dutch clog dancing, you as the dancer need to follow a very specific sequence of movements. The dance takes place on a square grid of square tiles, and at the start of the dance you stand on the top left corner tile of the grid. You then alternate between two types of dance move, moving from tile to tile in the grid for as long as you want. Your first move may be of either kind, but after that you need to strictly alternate between the two kinds of moves.

Both moves are similar to knight moves in chess: in the first type of move, you go from your current square to a square that is a tiles away along one axis of the grid and b tiles away along the other axis. Similarly, in the second type of move, you need to move c and d tiles along the respective axes. As you can freely swap the two axes and choose the movement direction along each axis, there can be up to 8 ways of performing a given type of move. Figure K.1 shows an example dance routine with $(a, b) = (1, 2)$ and $(c, d) = (2, 3)$.

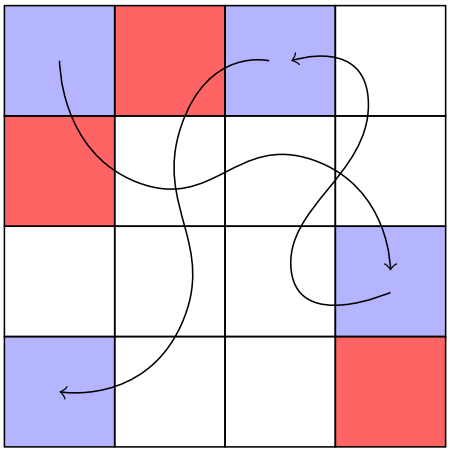


Figure K.1: Illustration of Sample Input 3, showing a dance that begins in the top left corner of a 4×4 grid and ends in the bottom left corner, visiting the blue squares along the way. There are 13 reachable squares in total. The three squares highlighted in red cannot be part of any dance performance.

Starting on the top left corner tile, how many different tiles could you reach while doing a clog dance? It is not allowed to step outside of the grid and you do not count tiles that you are simply stepping over while doing a move. Note that you need to count all tiles that can be reached during *some* performance of the dance, but not necessarily during the same one.

Input

The input consists of:

- One line with an integer n ($3 \leq n \leq 500$), the side length of the square.
- One line with two integers a and b ($1 \leq a, b < n$), describing the first dance move.
- One line with two integers c and d ($1 \leq c, d < n$), describing the second dance move.

Output

Output the number of tiles you can reach using these dance moves.

Sample Input 1

```
3
2 1
2 2
```

Sample Output 1

```
6
```

Sample Input 2

```
8
1 2
1 2
```

Sample Output 2

```
64
```

Sample Input 3

```
4
1 2
2 3
```

Sample Output 3

```
13
```

Sample Input 4

```
5
1 2
2 3
```

Sample Output 4

```
25
```

Sample Input 5

```
10
3 3
4 4
```

Sample Output 5

```
50
```

Problem L
Lateral Damage
Time limit: 2 seconds

You are playing *Battleships* in a large ocean with large ships. More precisely, there is a large square grid of size at most 100×100 and inside it are up to 10 of the largest type of ship in Battleships – the aircraft carrier – which has a length of five tiles, placed either horizontally or vertically. The ships do not overlap, but they are allowed to be adjacent to each other. See Figure L.1 for an example.

The original *Battleships* game, before the upgrade to a 100×100 grid.

Unfortunately, your opponent appears to bend the rules to their liking. It looks like they do not always determine the placement of their ships before you start shooting. You are not impressed by their attempt at cheating, and decide to try and win the game anyway.

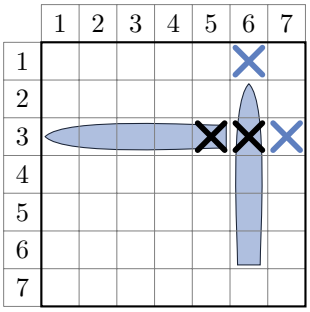


Figure L.1: Illustration of Sample Interaction 1 after the first four shots were fired.

Your goal is to locate and sink all your opponent's aircraft carriers in at most 2500 shots, that is, you must hit each of the five tiles of all their ships.

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:

The interactor first sends one line with two integers n and k ($5 \leq n \leq 100$, $1 \leq k \leq 10$), the size of the grid and the number of ships. It is guaranteed that it is possible to place k aircraft carriers in the grid without overlap.

Then, your program needs to start firing shots. Each shot is fired by printing one line of the form "x y" ($1 \leq x, y \leq n$), indicating you shoot at position (x, y). The interactor will respond with "`hit`" if the shot was a hit, "`sunk`" if the shot caused an aircraft carrier to sink, and "`miss`" otherwise. If you have shot the same location before, the response will be "`miss`".

Once you sink the last aircraft carrier, the interaction will stop and your program must exit.

The interactor is adaptive: the positions of the ships may be determined during the interaction, and may depend on where you decide to shoot.

Make sure you flush the buffer after each write.

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

Firing more than 2500 shots will result in a wrong answer.

Read	Sample Interaction 1	Write
`7 2`		
		`6 1`
`miss`		
		`6 3`
`hit`		
		`7 3`
`miss`		
		`5 3`
`hit`		
		`4 3`
`hit`		
		`3 3`
`hit`		
		`2 3`
`hit`		
		`1 3`
`sunk`		
		`6 7`
`miss`		
		`6 7`
`miss`		
		`6 2`
`hit`		
		`6 2`
`miss`		
		`6 4`
`hit`		
		`6 5`
`hit`		
		`6 6`
`sunk`		