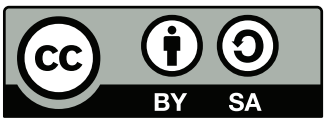
BAPC 2016

The 2016 Benelux Algorithm Programming Contest

Benelux Algorithm Programming Contest 2016

Problems

A	Airport Logistics
B	Battle Simulation
C	Brexit
D	Bridge Automation
E	Charles in Charge
F	Endless Turning
G	Manhattan Positioning System
H	Multiplying Digits
I	Older Brother
J	Programming Tutors
K	Safe Racing
L	Sticky Situation



A Airport Logistics

Many airports have moving conveyor belts in the corridors between halls and terminals. Instead of walking on the floor, passengers can choose to stand on a conveyor or, even better, walk on a conveyor to get to the end of the corridor much faster.

The brand new Delft City Airport uses a similar system. However, in line with the latest fashion in airport architecture, there are no corridors: the entire airport is one big hall with a bunch of conveyor lines laid out on the floor arbitrarily.

To get from a certain point A to a certain point B, a passenger can use any combination of walking on the floor and walking on conveyors. Passengers can hop on or off a conveyor at any point along the conveyor. It is also possible to cross a conveyor without actually standing on it.

Walking on the floor goes at a speed of 1 meter/second.
Walking forward on a conveyor goes at a total speed of 2 meter/second.
Walking in reverse direction on a conveyor is useless and illegal, but you may walk on the floor immediately next to the conveyor. (Conveyors are infinitely thin.)

How fast can you get from A to B?

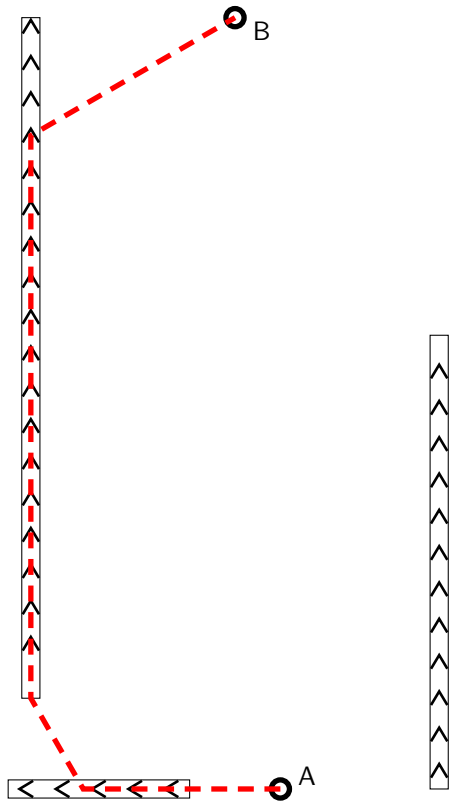


Figure 1: Fastest route for first example input.

Input

The first line contains four floating point numbers, X_A, Y_A, X_B, and Y_B. They describe the coordinates of your initial location $A = (X_A, Y_A)$ and your final location $B = (X_B, Y_B)$.

The second line contains an integer N, the number of conveyors in the hall ($0 \leq N \leq 100$). The following N lines each contain four floating point numbers, X_1, Y_1, X_2, and Y_2, describing a conveyor which starts at the point (X_1, Y_1) and ends at the point (X_2, Y_2), running in a straight line from start to end.

All coordinates are floating point numbers in the range $(0 \leq X, Y \leq 1000.0)$, expressed in units of meters.

Conveyors are at least 1 meter long. Conveyors do not intersect or touch.
Your start and destination are not on any conveyor.

Output

Write one line with a floating point number, the minimum time (in seconds) needed to get from A to B in seconds.

Your answer may have an absolute error of at most 10^{-4}.

Sample Input 1	**Sample Output 1**
60.0 0.0 50.0 170.0 3 40.0 0.0 0.0 0.0 5.0 20.0 5.0 170.0 95.0 0.0 95.0 80.0	168.791651

Sample Input 2	**Sample Output 2**
60.0 0.0 50.0 170.0 3 40.0 0.0 0.0 0.0 5.0 20.0 5.0 170.0 95.0 0.0 95.0 100.0	163.527474

Sample Input 3	**Sample Output 3**
0.0 1.0 4.0 1.0 1 0.0 0.0 4.0 0.0	3.732051

Sample Input 4	**Sample Output 4**
0.0 1.0 10.0 1.0 2 1.0 0.0 2.0 3.0 6.0 1.0 4.0 1.0	10.000000

B Battle Simulation

Picture by Bandai Namco via Wikimedia Commons

A terrible monster is rampaging through Neo Tokyo 5! The Earth Defense Force (EDF) has sent a mech unit[1] to defeat the monster. Because there is only a single mech unit available after previous monster rampages, the EDF has decided to simulate the upcoming battle between the mech and the monster before launching an assault. The EDF noted that the monster's attack pattern can be simulated by a series of moves that it performs in succession. When denoting each of its moves with a single letter, the attack pattern can be simulated as a single string, which should be read from left to right. The monster has the following moves:

- Rake, denoted by the letter '`R`';
- Bite, denoted by the letter '`B`';
- Laser breath, denoted by the letter '`L`'.

In order to defeat the monster, the mech must perform a counter move per move that the monster makes:

- Slice, denoted by the letter '`S`', counters the monster's rake;
- Kick, denoted by the letter '`K`', counters the monster's bite;
- Shield, denoted by the letter '`H`', counters the monster's laser breath;

However, there is one catch. When the monster performs a subsequent combination of the three moves *Rake*, *Bite* and *Laser breath*, in any order, it becomes a very powerful attack for which the mech must perform a single counter move called *Combo breaker*, denoted by the letter '`C`'. A single *Combo breaker* absorbs the entire combination of three moves. Any following moves from the monster will have to be countered separately or as part of a new combination. A move of the monster can never be part of more than one combination.

Through extensive analysis of the monster's past behaviour, the EDF is now able to reliably predict the actions of the monster ahead of time. You are given a string representing the moves that the monster will use when battling the mech. The EDF needs you to write a program that outputs the sequence of moves that the mech must perform in order to defeat the monster.

Input

A single line containing a string of at least 1 and at most 1 000 000 characters, consisting of the letters '`R`', '`B`' and '`L`'.

[1] huge bipedal robot, piloted by Japanese teenagers.

Output

Output a single string consisting of the letters denoting the moves that are to be made in succession by the mech in order to defeat the monster.

Sample Input 1	**Sample Output 1**
RRBBBLLR	SSKKKHHS

Sample Input 2	**Sample Output 2**
RBLLLBRR	CHCS

Sample Input 3	**Sample Output 3**
RBLBR	CKS

C Brexit

A long time ago in a galaxy far, far away, there was a large interstellar trading union, consisting of many countries from all across the galaxy. Recently, one of the countries decided to leave the union. As a result, other countries are thinking about leaving too, as their participation in the union is no longer beneficial when their main trading partners are gone.

Picture by NASA

You are a concerned citizen of country X, and you want to find out whether your country will remain in the union or not. You have crafted a list of all pairs of countries that are trading partners of one another. If at least half of the trading partners of any given country Y leave the union, country Y will soon follow. Given this information, you now intend to determine whether your home country will leave the union.

Input

The input starts with one line containing four space separated integers C, P, X, and L. These denote the total number of countries ($2 \leq C \leq 200\,000$), the number of trading partnerships ($1 \leq P \leq 300\,000$), the number of your home country ($1 \leq X \leq C$) and finally the number of the first country to leave, setting in motion a chain reaction with potentially disastrous consequences ($1 \leq L \leq C$).

This is followed by P lines, each containing two space separated integers A_i and B_i satisfying $1 \leq A_i < B_i \leq C$. Such a line denotes a trade partnership between countries A_i and B_i. No pair of countries is listed more than once.

Initially, every country has at least one trading partner in the union.

Output

For each test case, output one line containing either "`leave`" or "`stay`", denoting whether you home country leaves or stays in the union.

Sample Input 1

```
4 3 4 1
2 3
2 4
1 2
```

Sample Output 1

```
stay
```

Sample Input 2

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

Sample Output 2

```
leave
```

Sample Input 3

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

Sample Output 3

```
stay
```

Sample Input 4

```
10 14 1 10
1 2
1 3
1 4
2 5
3 5
4 5
5 6
5 7
5 8
5 9
6 10
7 10
8 10
9 10
```

Sample Output 4

```
leave
```

D Bridge Automation

In Delft there are a number of bridges that are still being operated by a human, known as the bridge operator. One such bridge operator will soon retire, hence there is the need for a replacement. The Bridge And Poker Committee has decided to use a computer program to automatically open and close the bridge, eliminating the need for human interaction.

However, the computer program still needs to be written. The requirements for this project are as follows:

1. No boat may be forced to wait for more than 30 minutes.
2. The amount of time during which the bridge is unavailable to road traffic must be as small as possible while still satisfying requirement 1.

It takes 60 seconds to raise or lower the bridge. During this time the bridge is not available to either road traffic or water traffic.

Boats arrive at the bridge at predictable times. It takes 20 seconds for a boat to sail through the bridge, assuming the bridge is already fully raised.

If the bridge is not fully raised when a boat arrives, the boat must wait. If there are boats waiting when the bridge becomes fully raised, these boats pass through the bridge one-by-one, which takes 20 seconds per boat. The bridge must remain fully raised as long as there are still boats sailing through! As soon as all boats have passed, the bridge may be lowered. But it might be more efficient to keep the bridge raised for a little while longer if the next boat is soon to arrive.

Given the arrival times of all boats, operate the bridge such that all boats can pass through without any boat waiting longer than 30 minutes. What is the total amount of time during which the bridge is unavailable to road traffic?

Input

The first line contains an integer N, the number of boats that must pass the bridge ($1 \leq N \leq 4\,000$).

Then follow N lines, each containing an integer T_i, the time at which boat i will arrive at the bridge in seconds ($60 \leq T_i \leq 10^5$).

Boats are sorted by increasing time of arrival, and never arrive within 20 seconds of each other ($i < j$ implies $T_i + 20 \leq T_j$).

Output

Write one line with an integer, the total number of seconds during which the bridge must be unavailable for road traffic in order for all boats to pass the bridge.

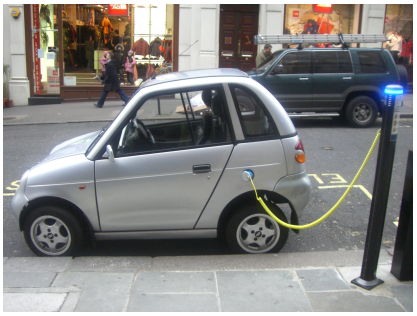
Sample Input 1	**Sample Output 1**
2 100 200	160

Sample Input 2	**Sample Output 2**
3 100 200 2010	250

Sample Input 3	**Sample Output 3**
3 100 200 2100	300

E Charles in Charge

Every day, Charles drives from his home to work and back. He uses the highways of the country that run from one city to another. Charles has decided that he wants to help the environment by buying an electrical car. Electrical cars, however, are not very common in his country yet. They can only be charged inside a city; there are no charging stations along the highways in between the cities. Moreover, all electrical cars are identical except for one thing: the size of the battery. As batteries are very expensive, Charles would like to buy a car with battery that is as small as possible.



Picture by Frank Hebbert via Flickr

However, this greatly increases the time it takes for him to get home, much to the distaste of his wife, Charlotte. This has spawned an argument, and after much discussion they have decided to compromise: Charlotte is fine with Charles taking a longer route, as long as it its length is at most $X\%$ longer than the length of shortest route that Charles could have taken to get home from work by using a regular car. Charles has agreed with this, and he now wants to find a route that minimizes the size of the car battery that he needs, i.e. the route that minimizes the maximum distance that Charles has to drive on a highway without passing through a city.

The amount of time Charles spends to charge his car can be neglected.

Input

The input starts with integers $2 \leq N \leq 10\,000$, $1 \leq M \leq 100\,000$ and $0 \leq X \leq 10\,000$: the number of cities, the number of highways connecting the cities and the aforementioned percentage X. City 1 is the place where Charles lives and city N is where he works.

Then follow M lines with on each line three integers: $1 \leq C_1 \leq N$, $1 \leq C_2 \leq N$, $1 \leq T \leq 10^9$. This means that there is a highway of length T connecting cities C_1 and C_2 (Charles can traverse the highway in both directions) *without* passing through any other cities. You may assume that there exists a path from city 1 to city N.

Output

The output is a single integer: the smallest maximum distance that Charles has to travel on a highway without passing through a city, such that the route he takes is at most $X\%$ longer than the shortest route.

Sample Input 1	Sample Output 1
2 1 100 1 2 5	5

Sample Input 2	Sample Output 2
9 8 15 1 9 16 1 4 4 4 5 4 5 6 4 6 8 4 4 7 5 7 8 5 8 9 4	5

Sample Input 3	Sample Output 3
9 8 30 1 9 16 1 4 4 4 5 4 5 6 4 6 8 4 4 7 5 7 8 5 8 9 4	4

Explanation of samples

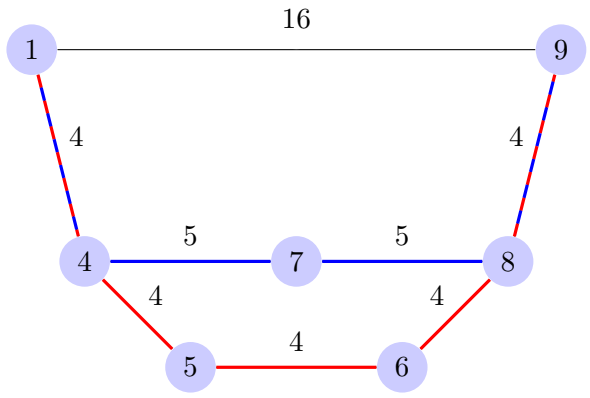


Figure 2: The graph of the second and third test case. The shortest path has length 16. In the second testcase, Charles's travel distance may not exceed $16 \cdot 1.15 = 18.4$ distance units. As a result, he cannot use a battery with which he can travel 4 distance units, as the red path 1–4–5–6–8–9 has length 20. Therefore, he uses the blue path 1–4–7–8–9, which has length 18, and the longest edge has length 5. In the third test case, he is allowed to travel $16 \cdot 1.30 = 20.8$ distance units, so he can follow the red path, where the longest edge has length 4.

F Endless Turning

Last week your little sister celebrated her birthday, and she was very pleased with her birthday present: a brand new scooter! Since then several times a day she goes for a drive, without telling anyone. She will leave the house, and go right on the pavement along the road. Fortunately she knows that she is not allowed to cross the road, and she is not sufficiently skilful with her new toy to turn around on the pavement. This means that, in order to continue her way, at each intersection she must turn right.

Every time your sister sets out, you are sent after her, of which you grow tired. Thus, knowing your little sister's ways with her scooter, you decide to write a program to find her.

Picture by Cha già José via Flickr

The city consists of R roads. Each road has a name which does not contain any spaces, and is an infinite line in the Euclidean plane. No three roads go through one point, i.e. at every intersection exactly two roads intersect. You figured out that your sister by this time should have completed N turns on the intersections of the city, unless of course she managed to leave the city by travelling on a road in a direction in which there is no other intersection, in which case she might not be able to even complete N turns. Your task is to write a program that tells you on which road your sister is right now.

Input

The first line contains four integers: the number of roads R, the number of turns N and the X- and Y-coordinate of your parents' home, satisfying $R \leq 100$, $N \leq 10^{10}$ and $|X|, |Y| \leq 10^7$.

The next R lines each describe a street. Each line contains one string S (without spaces, containing only alphanumeric characters, of length at most 20), the name of the street, and four integers X_1, Y_1, X_2, and Y_2, satisfying $|X_1|, |X_2|, |Y_1|, |Y_2| \leq 10^7$ and $(X_1, Y_1) \neq (X_2, Y_2)$, indicating that road S goes through points (X_1, Y_1) and (X_2, Y_2).

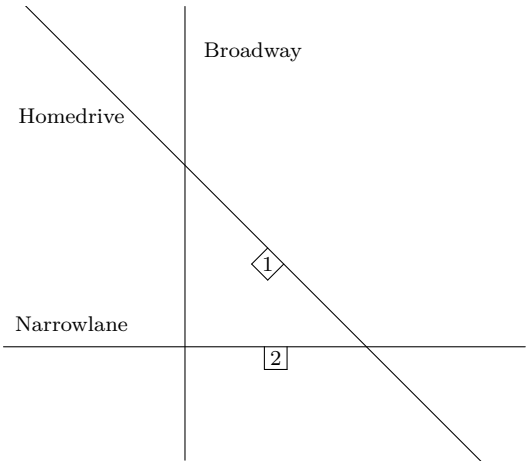
The location of your parents' home (X, Y) is guaranteed to lie on a unique non-vertical street (i.e. not on an intersection), on the south (negative Y-direction) side of the street, meaning that your little sister will depart in the east (positive X) direction. Moreover, each intersection is guaranteed to have coordinates of absolute value at most 10^7, and is guaranteed to lie at least 10^{-4} from each other intersection in the same street.

Output

Output a single line containing the name of the road where your little sister can be found.

Sample Input 1	**Sample Output 1**
3 4 1 1 Broadway 0 0 0 1 Narrowlane 0 0 1 0 Homedrive 1 1 2 0	Narrowlane

Sample Input 2	**Sample Output 2**
3 4 1 0 Broadway 0 0 0 1 Narrowlane 0 0 1 0 Homedrive 1 1 2 0	Homedrive



In the first case your sister starts on Homedrive heading east, turns right on Narrowlane, Broadway, Homedrive and Narrowlane, which means that after four turns she is in Narrowlane.

In the second case your sister starts on Narrowlane heading east and turns right on Homedrive, leaving the city behind her. She will never finish four turns and ends on Homedrive.

G Manhattan Positioning System

The Manhattan Positioning System (MPS) is a modern variant of GPS, optimized for use in large cities. MPS assumes all positions are discrete points on a regular two-dimensional grid. Within MPS, a position is represented by a pair of integers (X,Y).

To determine its position, an MPS receiver first measures its distance to a number of beacons. Beacons have known, fixed locations. MPS signals propagate only along the X and Y axes through the streets of the city, not diagonally through building blocks. When an MPS receiver at (X_R,Y_R) measures its distance to a beacon at (X_B,Y_B), it thus obtains the Manhattan distance: $|X_R - X_B| + |Y_R - Y_B|$.

Given the positions of a number of beacons and the Manhattan-distances between the receiver and each beacon, determine the position of the receiver. Note that the receiver must be at an integer grid position (MPS does not yet support fractional coordinates).

Figure 3: MPS is ideal for this city

Input

The first line contains an integer N, the number of beacons ($1 \leq N \leq 1000$). Then follow N lines, each containing three integers, X_i, Y_i, and D_i, such that $-10^6 \leq X_i, Y_i \leq 10^6$ and $0 \leq D_i \leq 4 \cdot 10^6$. The pair (X_i, Y_i) denotes the position of beacon i, while D_i is the Manhattan distance between receiver and beacon i.

No two beacons have the same position.

Output

If there is exactly one receiver position consistent with the input, write one line with two integers, X_R and Y_R, the position of the receiver.

If multiple receiver positions are consistent with the input, write one line with the word "`uncertain`".

If no receiver position is consistent with the input, write one line with the word "`impossible`".

Sample Input 1	**Sample Output 1**
3 999999 0 1000 999900 950 451 987654 123 13222	1000200 799

Sample Input 2

```
2
100 0 101
0 200 199
```

Sample Output 2

```
uncertain
```

Sample Input 3

```
2
100 0 100
0 200 199
```

Sample Output 3

```
impossible
```

Sample Input 4

```
2
0 0 5
10 0 6
```

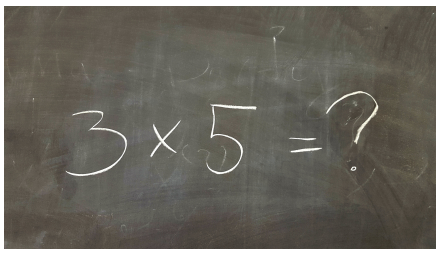
Sample Output 4

```
impossible
```

H Multiplying Digits

For every positive integer we may obtain a non-negative integer by multiplying its digits. This defines a function f, e.g. $f(38) = 24$.

This function gets more interesting if we allow for other bases. In base 3, the number 80 is written as 2222, so: $f_3(80) = 16$.



Picture by Mees de Vries

We want you to solve the reverse problem: given a base B and a number N, what is the smallest positive integer X such that $f_B(X) = N$?

Input

The input consists of a single line containing two integers B and N, satisfying $2 < B \leq 10\,000$ and $0 < N < 2^{63}$.

Output

Output the smallest positive integer solution X of the equation $f_B(X) = N$. If no such X exists, output the word "`impossible`". The input is carefully chosen such that $X < 2^{63}$ holds (if X exists).

Sample Input 1	**Sample Output 1**
`10 24`	`38`

Sample Input 2	**Sample Output 2**
`10 11`	`impossible`

Sample Input 3	**Sample Output 3**
`9 216`	`546`

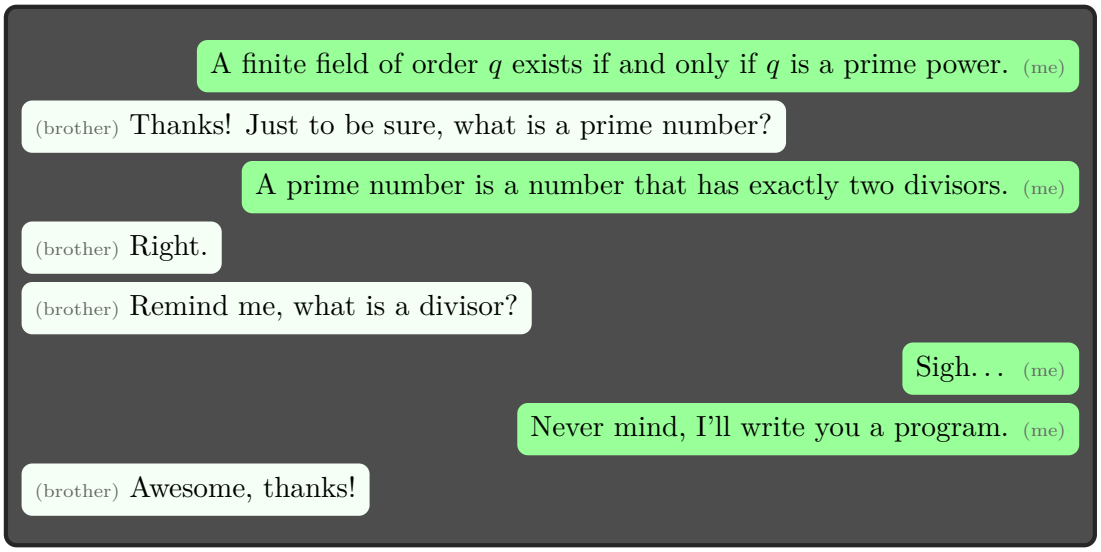
Sample Input 4	**Sample Output 4**
`10000 5810859769934419200`	`5989840988999909996`

This is not a blank page.

I Older Brother

Your older brother is an amateur mathematician with lots of experience. However, his memory is very bad. He recently got interested in linear algebra over finite fields, but he does not remember exactly which finite fields exist. For you, this is an easy question: a finite field of order q exists if and only if q is a prime power, that is, $q = p^k$ holds for some prime number p and some integer $k \geq 1$. Furthermore, in that case the field is unique (up to isomorphism).

The conversation with your brother went something like this:



Input

The input consists of one integer q, satisfying $1 \leq q \leq 10^9$.

Output

Output "`yes`" if there exists a finite field of order q. Otherwise, output "`no`".

Sample Input 1	**Sample Output 1**
`1`	`no`

Sample Input 2	**Sample Output 2**
`37`	`yes`

Sample Input 3	**Sample Output 3**
`65536`	`yes`

This is not a blank page.

J Programming Tutors

You are the founder of the Bruce Arden Programming Collective, which is a tutoring programme that matches experienced programmers with newbies to teach them. You have N students and N tutors, but now you have to match them up. Since the students will have to travel to their tutors' houses from their own (or vice versa) you decide to do your matching based on travel distance.

Picture by Damien Pollet via Flickr

Minimising overall distance doesn't seem fair; it might happen that one student has to travel a huge distance while all the other students get a tutor very close by, even though the tutors could have been split up so that each gets a tutor that is at least somewhat close.

Thus, you opt to minimise the distance travelled by the student who is worst off; one pairing of students to tutors is better than another if the student who has to travel farthest in the first pairing has to travel less far than the student who has to travel farthest in the second pairing.

Because the students live in a city, the distance that a student needs to travel is not the literal distance between them and their tutor. Instead, the distance between points (X, Y) and (X', Y') in the city is

$$|X - X'| + |Y - Y'|.$$

Input

The first line of the input contains an integer N, with $1 \leq N \leq 100$, the number of students and the number of tutors to pair up.

Then, there are N lines, each with two space separated integers with absolute value at most 10^8, which give the locations of the N students.

These are followed by N lines, each with two space separated integers with absolute value at most 10^8, which give the locations of the N tutors.

Note that it is possible for students and/or tutors to have identical locations (they may share a house).

Output

Output a single line containing a single integer K, where K is the least integer such that there exists a pairing of students to tutors so that no pair has distance greater than K between them.

Sample Input 1

```
2
0 0
0 3
0 2
0 5
```

Sample Output 1

```
2
```

Sample Input 2

```
4
0 1
0 2
0 3
0 4
1 0
1 1
1 2
1 3
```

Sample Output 2

```
2
```

Sample Input 3

```
3
0 5
5 2
4 5
3 3
5 2
5 2
```

Sample Output 3

```
5
```

Sample Input 4

```
2
0 0
0 5
-1 4
8 3
```

Sample Output 4

```
10
```

K Safe Racing

Tomorrow is racing day. There will be yet another grand prix in yet another country. Beside the safety car, there are various other security measures in order to make sure that everybody is as safe as possible. Among these safety measures are the track marshals: special race officials standing along the track with an assortment of flags that they can use to signal various messages to the drivers. For instance, the yellow flag warns the drivers of a dangerous situation, and the blue flag is used to order a lapped car to make way for one of the faster cars.

Picture by Martin Pettitt via Flickr

Every marshal should be stationed in a so-called *marshal booth*, a kind of protected cage that is clearly visible from the race track. These booths are located at regular intervals of ten metres (one decametre) along the track. The track is circular and L decametres long and therefore contains exactly L booths.

Not every booth needs to be used. International racing regulations require that the distance between two consecutive marshals should be at most S decametres, meaning that every S consecutive booths should contain at least one marshal. The marshals are not responsible for waving the finish flag, so it is not required (but also not forbidden) to have a marshal at the start/finish.

This leaves you with many ways of assigning marshals to the various booths along the track. Out of sheer curiosity you decide to calculate the total number of valid marshal assignments. Reduce your answer modulo 123 456 789 in case it gets too large.

Input

The input consists of two integers L, the length of the track, and S, the maximal distance between consecutive marshals along the track, satisfying $1 \leq S \leq L \leq 10^6$.

Output

Output the integer W, the number of ways to put marshals modulo 123 456 789. (Your answer must satisfy $0 \leq W < 123\,456\,789$.)

Sample Input 1	**Sample Output 1**
3 2	4

Sample Input 2	**Sample Output 2**
2500 2000	27511813

In the first sample test case, the four solutions are to put marshals at distances 0 and 1, at distances 0 and 2, at distances 1 and 2, or, at distances 0, 1 and 2 (in decametres) from the start.

This is not a blank page.

L Sticky Situation

Picture by Jeanette Irwin via Flickr

While on summer camp, you are playing a game of hide-and-seek in the forest. You need to designate a "safe zone", where, if the players manage to sneak there without being detected, they beat the seeker. It is therefore of utmost importance that this zone is well-chosen.

You point towards a tree as a suggestion, but your fellow hide-and-seekers are not satisfied. After all, the tree has branches stretching far and wide, and it will be difficult to determine whether a player has reached the safe zone. They want a very specific demarcation for the safe zone. So, you tell them to go and find some sticks, of which you will use three to mark a non-degenerate triangle (i.e. with strictly positive area) next to the tree which will count as the safe zone. After a while they return with a variety of sticks, but you are unsure whether you can actually form a triangle with the available sticks.

Can you write a program that determines whether you can make a triangle with exactly three of the collected sticks?

Input

The first line contains a single integer N, with $3 \leq N \leq 20\,000$, the number of sticks collected. Then follows one line with N positive integers, each less than 2^{60}, the lengths of the sticks which your fellow campers have collected.

Output

Output a single line containing a single word: `possible` if you can make a non-degenerate triangle with three sticks of the provided lengths, and `impossible` if you can not.

Sample Input 1	**Sample Output 1**
`3` `1 1 1`	`possible`

Sample Input 2	**Sample Output 2**
`5` `3 1 10 5 15`	`impossible`

Benelux
Algorithm
Programming
Contest
2016

Solutions

BAPC 2016

Delft University of Technology

22 October 2016

A: Airport Logistics [1/3]

Benelux Algorithm Programming Contest 2016

The solution for this problem has two parts:

1. Create a directed graph, with nodes representing points on the floor and cost-labeled edges representing the time to walk from one point to another,
2. find the shortest path in that graph.

Part 2 can be done using Dijkstra's algorithm.
Part 1 is the hard part.

A: Airport Logistics [2/3]

Doing some geometry, we find the following rules:

- The optimal path consists of straight line segments.
- When an optimal path joins a conveyor halfway (i.e. not at the begin of the conveyor), this conveyor is approached via a straight line intercepting the conveyor at a 60-degree angle.
- When an optimal path leaves a conveyor halfway (i.e. not at the end of the conveyor), the path leaves the conveyor via a straight line at a 60-degree angle with the conveyor.
- It is never necessary to leave one conveyor halfway and join the next conveyor halfway.

A: Airport Logistics [3/3]

According to these rules we connect:

- the starting point with each belt,
- each belt to the end point,
- each pair of belts,
- (finally) the nodes within each belt - going from entrance nodes to exit nodes.

This graph has $O(N^2)$ nodes and $O(N^2)$ edges in the worst case. The shortest path in the graph is then found with Dijkstra's algorithm in time $O(E * log(E))$.

B: Battle Simulation

Benelux Algorithm Programming Contest 2016

Problem: replace characters in given string S. When subsequent combination occurs of all 3 characters, replace those 3 with one character instead.

B: Battle Simulation

Benelux Algorithm Programming Contest 2016

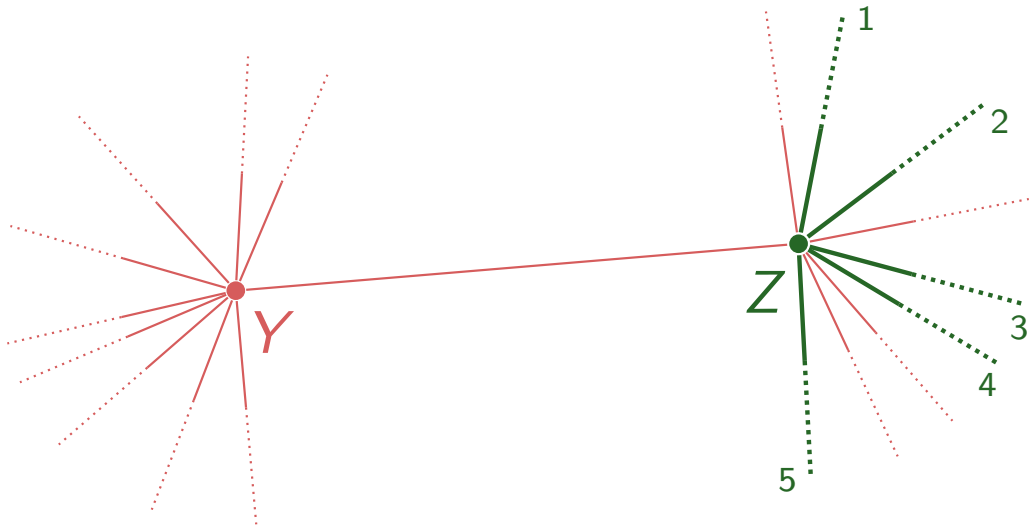
Problem: replace characters in given string S. When subsequent combination occurs of all 3 characters, replace those 3 with one character instead.

- Linearly replace all characters. Lookahead two characters to see if any combination occurs of three different characters.
- If so, ignore following two characters and continue replacing characters.
- Or... Use regular expressions instead! E.g. S.replaceAll("RBL—RLB—BRL—BLR—LRB—LBR", "C");
- String concatenation is too slow.

C: Brexit [1/2]

Benelux Algorithm Programming Contest 2016

- Simulation with some emphasis on efficiency.
- Look locally: when removing a country Y, see if this pushes one of its partners Z over the tipping point.
- Don't perform a recount every time we consider Z:

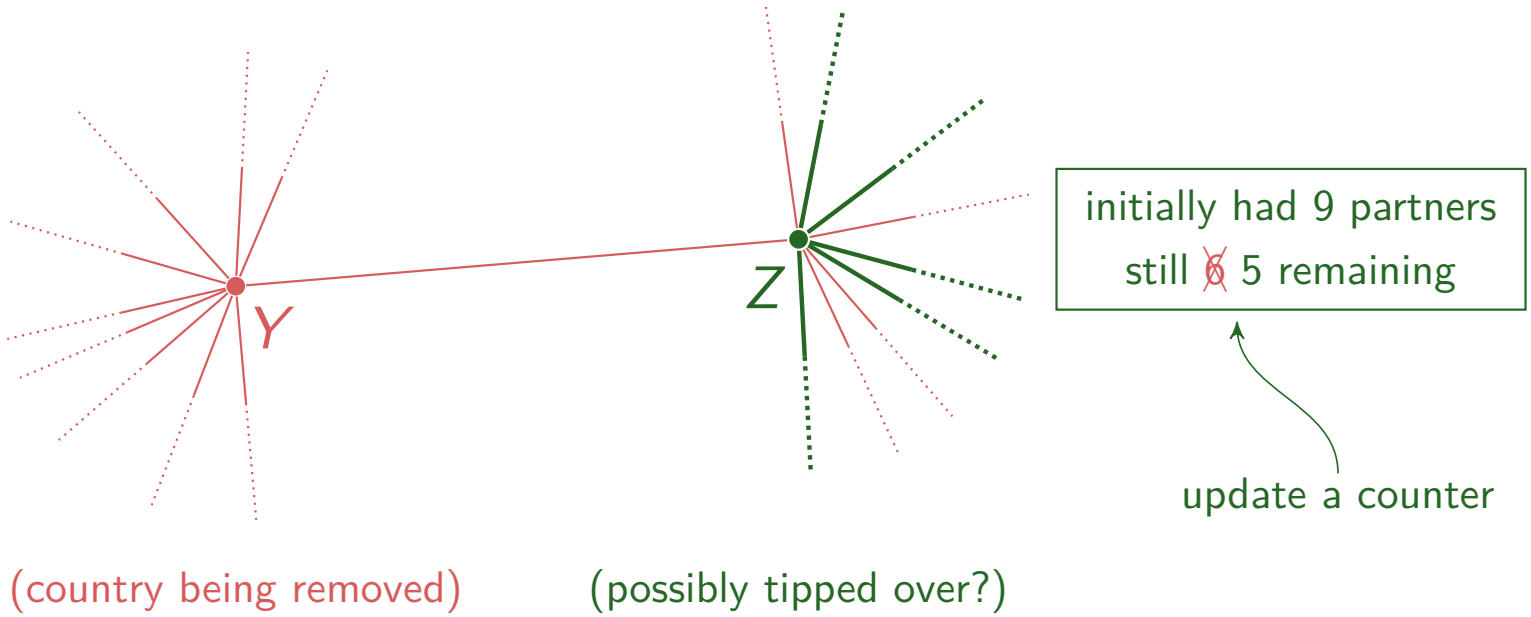


(country being removed) (possibly tipped over?)

Complexity can be up to $\Theta(P^2)$, which is too slow!

C: Brexit [2/2]

- Instead we keep count:



- Can be implemented breadth-first or depth-first.
- Time complexity: $\mathcal{O}(C + P)$.

D: Bridge Automation [1/2]

Benelux Algorithm Programming Contest 2016

Task:

- No boat may wait more than 1800 seconds.
- Minimize amount of time where bridge is not fully closed.

Strategy:

- Keep bridge closed until oldest boat has waited (1800 - 60) seconds.
- Then open bridge, let the next k boats through, then close it.
- Repeat until all boats passed.

D: Bridge Automation [2/2]

Benelux Algorithm Programming Contest 2016

Algorithm: dynamic programming

table$[p]$ = minimum cost needed to let the first p boats pass

table$[0] = 0$

table$[p] =$

$$\min_{1 \le k \le p} \Big(\text{table}[p-k] + \max\{ T_p - T_{p-k+1} - 1800 + 20, 20k \} + 120 \Big)$$

(Assume first $(p-k)$ boats already passed;
let boat $(p-k+1)$ wait exactly 1800 seconds, then open bridge;
keep bridge open until boat (p) has passed, then close bridge.)

Final answer is table$[n]$

E: Charles in Charge [1/2]

Benelux Algorithm Programming Contest 2016

Problem: given a graph G, find the lowest value such that the shortest path using only edges of length at most this lowest value is at most $X\%$ longer than the shortest path without any limitations.

E: Charles in Charge [1/2]

Benelux Algorithm Programming Contest 2016

Problem: given a graph G, find the lowest value such that the shortest path using only edges of length at most this lowest value is at most $X\%$ longer than the shortest path without any limitations.

Some notation:

- let $G = (V, E)$ be the given graph and let D be the maximum distance Charles is allowed to travel;
- for a value K, let $G_k = (V, E_K)$ be the subgraph of G using only edges of length at most K;
- let D_K be the shortest distance from 1 to N in G_K.

E: Charles in Charge [2/2]

Benelux Algorithm Programming Contest 2016

The problem is now formulated as follows: what is the smallest K such that the shortest path from 1 to N in G_K is at most D?

E: Charles in Charge [2/2]

Benelux Algorithm Programming Contest 2016

The problem is now formulated as follows: what is the smallest K such that the shortest path from 1 to N in G_K is at most D?

We make a pair of observations:

1. Given a value K, we can calculate the shortest path from 1 to N in G_K using Dijkstra.
2. For any value $L \geq K$ we have $D_L \leq D_K$.

E: Charles in Charge [2/2]

Benelux Algorithm Programming Contest 2016

The problem is now formulated as follows: what is the smallest K such that the shortest path from 1 to N in G_K is at most D?

We make a pair of observations:

1. Given a value K, we can calculate the shortest path from 1 to N in G_K using Dijkstra.
2. For any value $L \geq K$ we have $D_L \leq D_K$.

Hence we can use *binary search* to find the correct value of K and solve the problem.
Runtime: $O(|E| \log(|V|) \log(|E|))$.

F: Endless Turning

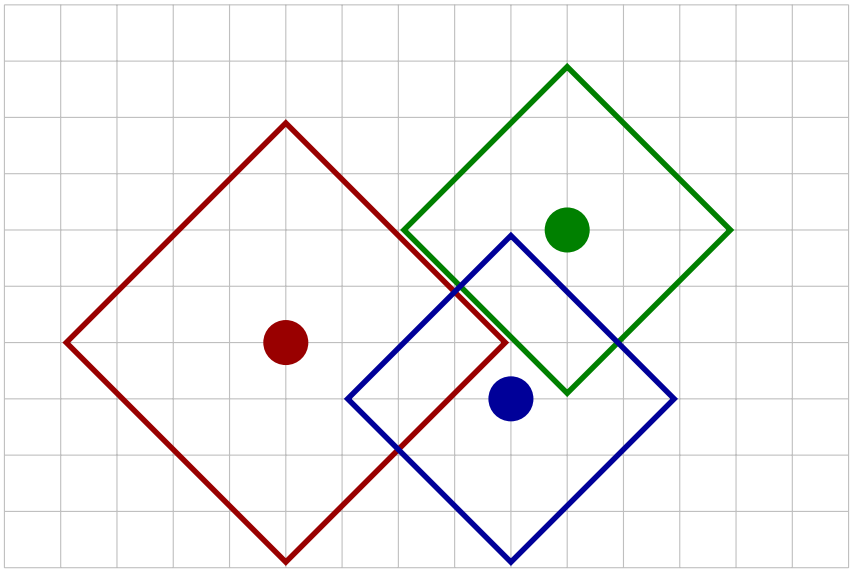
Benelux Algorithm Programming Contest 2016

- For each pair of streets calculate their intersection.
- For each street find the order in which the intersection points lie in that street, using a sort algorithm.
- Find the street on which the starting point is located.
- Now simulate the driving, keeping track of the direction in which you are traversing the streets.
- If you arrive at the first intersection for the second time, take N modulo the number of turns taken so far.
- Finish the simulation.
- *Funny fact:* as you walk around a polygon, in each street you will visit only two intersections: one where you enter each time and one where you leave.

G: Manhattan Positioning System [1/2]

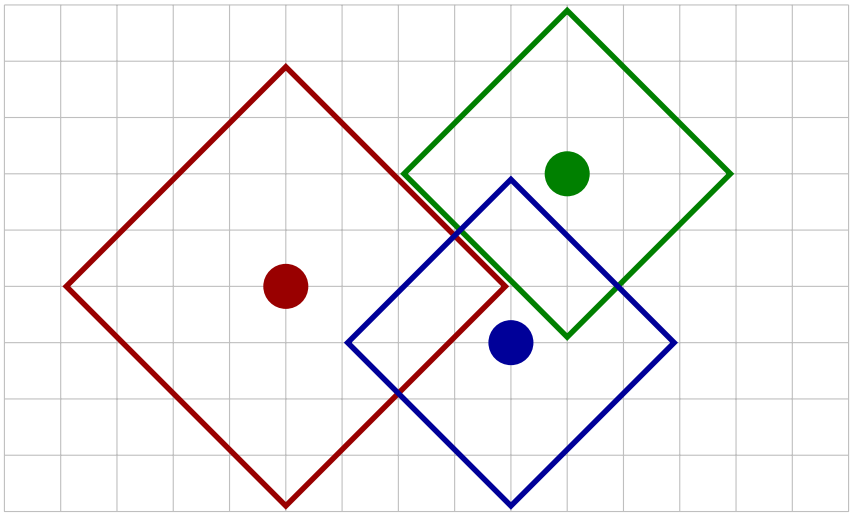
Benelux Algorithm Programming Contest 2016

- Task: Find a unique point at specific Manhattan distance to each beacon.
- The set of points at specific distance to one beacon is a "circle". Under Manhattan distance metric, a "circle" looks like a diamond shape.
- Task: Find the intersection of the diamond shapes of all beacons.



G: Manhattan Positioning System [2/2]

- Choose one beacon.
- Create abstract representation of its diamond shape:
  Set of line segments, $\{(x1, y1, x2, y2), \ldots\}$.
- Visit all other beacons, and intersect the remaining set of line segments with the other beacon's diamond shape.
- After processing all beacons; the set of line segments is empty (*impossible*) or contains exactly one point (unique solution), or contains multiple points/segments (*uncertain*).



H: Multiplying Digits [1/3]

Benelux Algorithm Programming Contest 2016

Given a number n and a base b, find the least x such that the product of the digits of x (x written in base b) equals n.
Possible **digits** of x are divisors of n that are less than b.
Find an *ascending* sequence of digits, such that

- Their product equals n,
- the sequence is as short as possible
- the sequence is lexicographically minimal.

It is tempting to put the largest possible digit at the end. But that is wrong (Sample 3):

- $b = 9$; $n = 216 = 2 * 2 * 2 * 3 * 3 * 3$.
- Choosing 8 as last digit gives 3 3 3 8 $\Rightarrow$ 1115.
- However 6 6 6 $\Rightarrow$ 546 is a better (the best) solution.

H: Multiplying Digits [2/3]

Benelux Algorithm Programming Contest 2016

Dynamic Programming, memoize the function Best:

- If n has a prime divisor $\geq b$, there is no solution.
- function Best(long k) gives the best solution.
- base case: if (k $<$ b) Best $=$ k
- recursion:

```
for (d < BASE, d divides k)
    find solutions ending with digit d, as follows:
        k1 = k/d
        b1 = Best(k1)
        sol1 = b * b1 + d
and return the best (least) of the sol1
```

- The *least* of the sol1 will be less than 2^{63}, but not necessarily *all* sol1,
- so beware of overflow!

H: Multiplying Digits [3/3]

Unfortunately, this is not fast enough. We need some of the following optimizations:

- Store the possible digits beforehand (the divisors of n below b)
- If $d * d < b$ then d will not occur in an optimal solution, except as the *first* digit. The left neighbour of d, say d_1, is at most d so the two can be replaced by $d_1 * d < b$, making a smaller number.
- If a multiple of a digit d can be chosen as the last digit in the solution for some k, then d will not be the last digit in the optimal solution for that k.

I: Older Brother

Benelux Algorithm Programming Contest 2016

Is q a prime power? Use a simplified factorization algorithm:

```
1  bool isPrimePower(int q) {
2      if (q == 1) // Corner case.
3          return false;
4      for (int p = 2; p * p <= q; p++) {
5          if (q % p == 0) {
6              // Least divisor will be prime.
7              // Check if q is a power of p.
8              while (q % p == 0)
9                  q /= p;
10             return q == 1;
11         }
12     }
13     // Apparently, q is prime.
14     return true;
15 }
```

J: Programming Tutors

We are looking for a matching which minimizes the maximal distance between pairs. Some ways to solve this efficiently enough include:

- Use a binary search over the maximal distance. Given a candidate maximal distance, use your favourite matching algorithm.
- Use a standard minimal matching algorithm, but look for augmenting paths with minimal highest distance, instead of minimal total distance.
- Even fast enough: start with an empty partial matching, allow new edges one by one starting with the shortest, look for a new augmenting path each time.

K: Safe Racing [1/2]

Benelux Algorithm Programming Contest 2016

- General remark: reduce modulo 123456789 in all intermediate calculations to avoid overflow.
- Calculate

$$D[i] = \begin{cases} \text{number of ways to allocate marshalls to} \\ \text{booths 0 up to and including } i \text{ given that} \\ \text{there is a marshall in booths 0 and } i \end{cases}$$

for $i = 0, \ldots, L-1$, using dynamic programming in runtime $\mathcal{O}(L)$ using:

$$D[i] = \sum_{j=\max(0,i-S)}^{i-1} D[j].$$

- During the process, keep track of the partial sums of the last S values. Do not recalculate them to avoid getting runtime $\mathcal{O}(S \cdot L)$, which is too big.

K: Safe Racing [2/2]

- If the first marshall is at position f and the last one at position $L - g$ (satisfying $f \geq 0$, $g \geq 1$ and $f + g \leq S$), then the number of ways to put marshalls in between these positions is $D[L - f - g]$.
- Hence, the answer is

$$\sum_{f=0}^{S} \sum_{g=1}^{S-f} D[L - f - g],$$

but naively it would take $\mathcal{O}(S^2)$ time to calculate this.

- Notice that each value of $h := f + g$ occurs h times in the sum. Hence, we can also write the answer as

$$\sum_{h=1}^{S} D[L - h] \cdot h,$$

which can be calculated in $\mathcal{O}(S)$.

L: Sticks [1/2]

Benelux Algorithm Programming Contest 2016

- Among a sequence of numbers, are there three that form the side of a triangle?
- That is, are there $a < b < c$ with $a + b > c$?

L: Sticks [1/2]

Benelux Algorithm Programming Contest 2016

- Among a sequence of numbers, are there three that form the side of a triangle?
- That is, are there $a < b < c$ with $a + b > c$?
- There are too many to check all triples.
- If any triple works, then a triple of consecutive lengths does.
- Solution: sort the list of stick lengths. Check if sticks $i, i + 1, i + 2$ form a triangle.

L: Sticks [2/2]

- The biggest set of sticks for which no solution exists are Fibonacci numbers:

$$1, 1, 2, 3, 5, 8, 11, \ldots$$

- The largest Fibonacci number allowed ($< 2^{60}$) is F_{88}.

L: Sticks [2/2]

- The biggest set of sticks for which no solution exists are Fibonacci numbers:

$$1, 1, 2, 3, 5, 8, 11, \ldots$$

- The largest Fibonacci number allowed ($< 2^{60}$) is F_{88}.
- Silly solution: if $n > 90$, it is always possible.
- If $n \leq 90$, check all possible triples.