A 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

B 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
```

C 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
```

D 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).

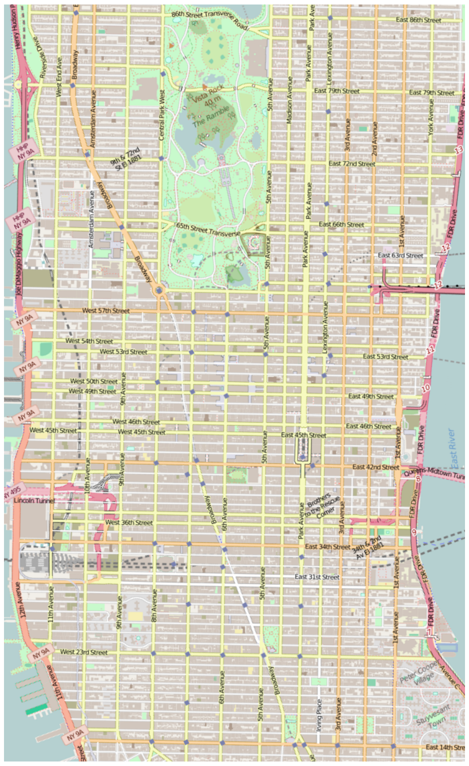


Figure 3: MPS is ideal for this city
© OpenStreetMap contributors

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
```

E 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

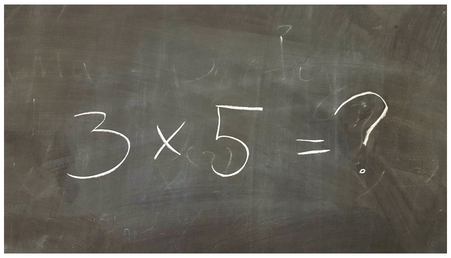
Output the integer W, the number of ways to put marshals modulo 123 456 789. (Your answer must satisfy $0 \leq W < 123\,456\,789$.)

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.

F Multiplying Digits



Picture by Mees de Vries

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: $f_3(80) = 16$.

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`