Benelux Algorithm Programming Contest

Finals

Saturday October 15th 2011

Contents

A Popping Balloons **3**

B Quick out of the Harbour **5**

C Find the Treasure **7**

D Bad Wiring **9**

E Undercover Pirate **11**

F Ultimate Finishing Strike **13**

G Doubloon Game **15**

H Walking the Plank **17**

I Parking Ships **19**

J Treasure Map **21**

A Popping Balloons

Problem

John loves programming contests. There is just one problem: his team is not very good at programming. This usually doesn't bother him, but what does bother him is that everyone gets a balloon for every correct submission. John's team never gets any balloons, while other teams get one balloon after the other. This frustrates him, so John would like to see that all other teams have no balloons either.

This year he has a plan to achieve just that. John has hired a ninja to pop all balloons for him. At any time during the contest, he can call for the ninja to come down through a hole in the ceiling and pop all balloons by using his shurikens (ninja stars), before leaving through the hole in the ceiling again. Of course the ninja wants to use as few of his precious shurikens as possible. Therefore, John must write a program that computes how many shurikens are needed to pop all balloons. Because all balloons are usually at approximately the same height, he can model the problem as a 2-dimensional problem. He sets the location of the ninja (where he comes in) as the origin $(0, 0)$ and uses circles to model the balloons. To be on the safe side, these circles can have different radii. Shurikens are assumed to be thrown from the origin and move in a straight line. Any circle/balloon crossed by this halfline will be popped by this shuriken. The question then becomes: how many halflines rooted at the origin are necessary to cross all circles?

Of course, as mentioned above, John is not a very good programmer, so he asks you to make this program for him. Can you help him out? You might get a balloon if you get it right...

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with a single integer n ($0 \leq n \leq 1,000$): the number of balloons.
- n lines, each containing three integers x_i, y_i ($-10^4 \leq x_i, y_i \leq 10^4$), and r_i ($1 \leq r_i \leq 10^4$), describing the circle used to model the i^{th} balloon, where (x_i, y_i) is the center of the circle and r_i is the radius.

You can assume that two lines (rooted at the origin) that are tangent to two distinct circles make an angle of at least 10^{-6} radians at the origin. Furthermore, the circles do not cross each other (but can touch) and do not contain the origin.

Output

For every test case in the input, the output should contain one integer on a single line: the minimum number of shurikens the ninja needs to pop all balloons.

Examples

Input	Output

2
5
2 0 1
5 0 2
0 3 2
-4 0 2
0 -2 1
5
4 1 3
5 -5 3
0 -4 2
-4 4 3
-10 3 3
	4
	3

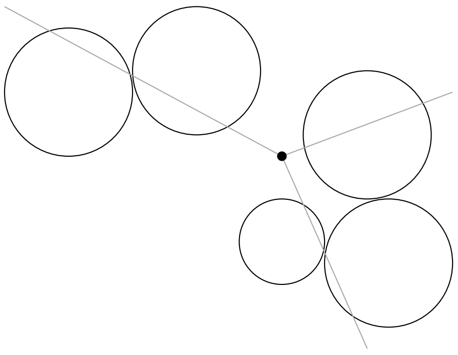


Figure 1: Second sample case

Disclaimer

No balloons were harmed during the making of this problem.

B Quick out of the Harbour

Problem

Captain Clearbeard decided to go to the harbour for a few days so his crew could inspect and repair the ship. Now, a few days later, the pirates are getting landsick[1]. Before all of the pirates become too sick to row the boat out of the harbour, captain Clearbeard decided to leave the harbour as quickly as possible.

Unfortunately the harbour isn't just a straight path to open sea. To protect the city from evil pirates, the entrance of the harbour is a kind of maze with drawbridges in it. Every bridge takes some time to open, so it could be faster to take a detour. Your task is to help captain Clearbeard find the fastest way out to open sea.

The pirates will row as fast as one minute per grid cell on the map. The ship can move only horizontally or vertically on the map. Making a 90 degree turn does not take any extra time.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with three integers, h, w ($3 \leq h, w \leq 500$), and d ($0 \leq d \leq 50$), the height and width of the map and the delay for opening a bridge.
- h lines with w characters: the description of the map. The map is described using the following characters:
  - 'S', the starting position of the ship.
  - '.', water.
  - '#', land.
  - '@', a drawbridge.

Each harbour is completely surrounded with land, with exception of the single entrance.

Output

For every test case in the input, the output should contain one integer on a single line: the travelling time of the fastest route to open sea. There is always a route to open sea. Note that the open sea is not shown on the map, so you need to move outside of the map to reach open sea.

[1]Pirates get landsick when they don't get enough of the ships' rocking motion. That's why pirates often try to simulate that motion by drinking rum.

Examples

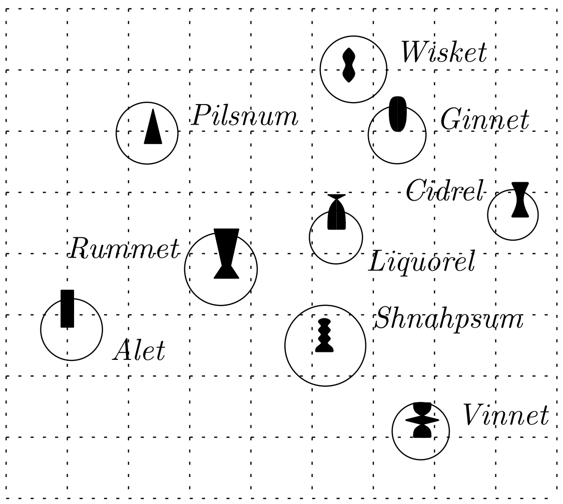
Input	Output

2
6 5 7
`#####`
`#S..#`
`#@#.#`
`#...#`
`#@###`
`#.###`
4 5 3
`#####`
`#S#.#`
`#@..#`
`###@#`
	16
	11

C Find the Treasure

Problem

Four hundred years ago, a group of pirates hid a treasure on an island in an archipelago that consists of many very small islands. Unfortunately, these pirates were particularly bad navigators and cartographers. Therefore, instead of a map, they made drawings of views from the top of the mountain on the treasure island. Each view shows two or more other islands of the archipelago that can be seen from the treasure island, ordered from left to right. The views also contain lots of fog, so the drawings may fail to show some islands that must have been in the field of view between the islands that appear in the drawing. For example, in Figure 2 below, if the treasure is hidden on Rummet, then the pirates could have drawn a view showing (from left to right) Wisket, Ginnet and Vinnet, or a view showing (from left to right) Liquorel and Cidrel. The pirates are known to have had acute vision, with a viewing angle of 180 degrees, but bad drawing skills: the distances between Wisket, Ginnet and Vinnet in their drawing are meaningless, and Liquorel may be drawn far to the left of Cidrel while in the actual view from Rummet, Liquorel must have been only slightly to the left of Cidrel, even obscuring part of it. Fortunately, all islands in the drawings can be identified easily thanks to the unique towers on top of each island.



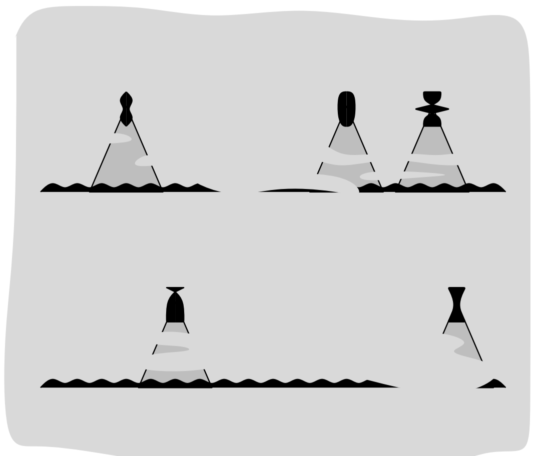


Figure 2: Map of a hypothetical archipelago, and some views from Rummet as the pirates could have drawn them.

Now, four hundred years later, you have got the pirates' drawings and an excellent, accurate map of the archipelago, showing all islands. On which island is the treasure hidden?

Input

The first line of the input contains a single number: the number of archipelagos to follow. Each archipelago has the following format:

- One line with an integer n, satisfying $1 \leq n \leq 125,000$: the number of islands in the archipelago.
- n lines, each with two integers x_i and y_i, satisfying $0 < x_i < 2^{29}$ and $0 < y_i < 2^{29}$: these are the coordinates of the tower T_i on each island.
- One line with an integer k: the number of test cases for this archipelago. Each test case has the following format:

– One line with an integer m, satisfying $0 \leq m \leq 10,000$: the number of pairs of islands that appear in the pirates' drawings.
– m lines, each with two integers l and r such that $1 \leq l \leq n$, $1 \leq r \leq n$, $l \neq r$, meaning that the tower T_l is drawn to the left of tower T_r.

In any archipelago, no two towers have the same x-coordinate, no two towers have the same y-coordinate, and no three towers lie on a line.

Output

For every test case in the input, the output should contain:

- One line with an integer i ($1 \leq i \leq n$), identifying the i-th island in the archipelago, for each island that could be the treasure island. These lines need to be in increasing order.
- One line containing the number 0.

Example

The following input describes one archipelago with one test case, namely the information corresponding to the map and the views shown in Figure 2. The potential treasure islands are Rummet, Alet, and Schnahpsum.

Input	Output
1	6
9	7
28 34	8
32 30	0
12 29	
27 22	
42 23	
18 18	
5 14	
26 12	
34 5	
1	
4	
1 2	
1 9	
2 9	
4 5	

D Bad Wiring

Problem

The ninja Ryu has infiltrated the Shadow Clan fortress and finds himself in a long hallway. Although ninjas are excellent fighters, they primarily rely on stealth to complete their missions. However, many lights are turned on in the hallway, and this way it will not take long before Ryu is spotted by a guard. To remain unseen, Ryu will need to turn off all the lights as quickly as possible.

The hallway contains a sequence of n lights $L_1 \ldots L_n$. Some of these lights are turned on. Destroying the lights with his shurikens would be too loud, so he needs to turn them off the old-fashioned way, using light switches. Luckily, there is a switch box nearby with a light switch S_i for every light L_i. However, after trying one of the switches, he notices something funny. When he flips the switch S_i, it does not only turn on/off light L_i, but also some of the neighboring lights. Ryu notices that there is a parameter D such that flipping switch S_i turns on/off all the lights $L_{i-D} \ldots L_{i+D}$, if they exist[2]. Turning on or off lights can attract the attention of the guards, so Ryu would like to turn off all the lights with the minimum number of times flipping a switch. Can you help him out?

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with two integers n ($1 \leq n \leq 100$) and D ($0 \leq D \leq 15$): the number of lights and the parameter mentioned above.
- One line with n integers. The i^{th} integer describes the current state of light L_i, where 0 means off and 1 means on.

Output

For every test case in the input, the output should contain one integer on a single line: the minimum number of times Ryu needs to flip a switch to turn off all the lights. If it is impossible to turn off all the lights, then output the string "`impossible`" instead.

Examples

In the first example below, flipping switch S_4 followed by S_7 will turn off all the lights.

Input	Output

2
7 3
1 1 1 0 0 0 0
5 1
1 0 1 0 1
	2
	3

[2]This means that S_1 turns on/off all the lights $L_1 \ldots L_{D+1}$ and S_n turns on/off all the lights $L_{n-D} \ldots L_n$. Of course, if $D \geq n$, then L_{D+1} and L_{n-D} will not exist either.

This page is intentionally left blank.

E Undercover Pirate

Problem

Panic in the dojo! An empty bottle of rum has been found inside the dojo of the Beat All Pirates Clan. This can mean only one thing: a pirate has gone undercover as a ninja of the clan! Clearly, this issue must be resolved immediately, so clan leader Hanzo gathers all ninjas in one place. The pirate has no intention to reveal himself, so Hanzo needs to come up with a clever way to expose the pirate.

It is well known that ninjas of the Beat All Pirates Clan are trained to have a perfect weight W. It is very unlikely that the pirate also has this weight[3], so Hanzo can expose the pirate by weighing all ninjas. Next to exposing him, Hanzo would also like to know whether the pirate is lighter or heavier than W, purely out of curiosity. However, weighing the ninjas one by one will take all day, so he needs a better way to do this. Luckily the clan owns a huge scale which can hold an arbitrary number of ninjas on each side. Hanzo knows that, for n ninjas including the pirate, he needs to use the scale only $\lceil \log_3(2n+3) \rceil$ times to expose the pirate, where $\lceil x \rceil$ is the smallest integer larger than or equal to x. Unfortunately he does not remember how to do it. Can you help him out?

Input and output

This is a problem with dynamic input and output, read the following description very carefully.

The first line of the input contains a single number: the number of test cases to follow. Then, for every test case you get one line with an integer n ($3 \leq n \leq 10^6$): the number of ninjas including the pirate. Ninjas are numbered from 1 to n. Then your program can make at most $\lceil \log_3(2n+3) \rceil$ queries.

Query: A query must be formatted as follows. You first describe the ninjas on the left side of the scale. This description can contain at most 4 ranges of numbers, written as $a - b$ (no spaces) separated by spaces[4]. Then follows a '`+`', followed by the description of the ninjas on the right side of the scale.

Answer: The answer of the query consists of a single character, either '`L`' (left side is heavier), '`R`' (right side is heavier) or '`E`' (both sides have the same weight).

After at most $\lceil \log_3(2n+3) \rceil$ queries your program must expose the pirate by outputting the string "`Ninja` i `is a heavy pirate`" or "`Ninja` i `is a light pirate`", depending on whether the pirate is heavier or lighter than W, where i is the number of the pirate. This ends the sequence of input and output for this test case.

If your program exceeds the number of allowed queries, or if your program outputs an invalid query (ranges are not disjoint and/or include ninjas outside of the range $[1, n]$), you will receive a Wrong Answer.

[3]Therefore you can assume that the pirate does not have weight W.

[4]Singleton ranges can be written as $a - a$ or simply a. Ranges $a - b$ where $b < a$ are handled as empty ranges. However, negative values for a or b are never allowed

Notes

C/C++ users:

- Do `fflush(NULL);` after you print output.
- If you use `scanf` to read the input, then do not read newline characters. So, to read an integer n with `scanf`, use `scanf("%d", &n);`.

Java/C# users:

- Do not use *buffered* output, `System.out.println`/`System.Console.WriteLine` will do fine.

In case you read the query answers as characters, note that every query answer is followed by a newline character.

Examples

Note that there might be multiple ways to find the correct answer, the sequence below is just an example. Extra newlines in the example below are for clarity only; you should print exactly one newline character after each line of output.

Input	Output
`3`	
`3`	
	`1 + 2`
`L`	
	`1 + 3`
`L`	
	`Ninja 1 is a heavy pirate`
`12`	
	`1-4 + 5-8`
`E`	
	`9-11 + 1-3`
`L`	
	`9 + 10`
`R`	
	`Ninja 10 is a heavy pirate`
`12`	
	`1-4 + 5-8`
`R`	
	`5-6 1 + 7-8 2`
`L`	
	`5 + 6`
`E`	
	`Ninja 2 is a light pirate`

F Ultimate Finishing Strike

Problem

As a ninja, Saito Hajime has to fight many opponents who are foolish enough to challenge his might. Most of these opponents fall easily to Saito's great martial arts techniques and ninjitsus[5]. From time to time however, the great Saito Hajime has to take care of a particularly powerful and skilled foe[6]. This foe usually enters the combat after several dozens of his/her minions have been defeated by Saito. Saito always encounters such foes in empty rectangular rooms.

In order to defeat such a powerful foe, Saito has to perform a special ninjitsu known as *Saito Hajime's Zero Stance Ultimate Finishing Strike*. This *strike* involves hitting his foe by performing a flying kick that starts at Saito's current position. Of course, a simple flying kick will not be enough to defeat a powerful foe, but Saito can improve the power of his strike by bouncing off several walls before hitting his foe. Every bounce gives his attack more power, so that with enough bounces any foe can be defeated. Note that Saito always bounces off a wall according to the rule "angle of incidence is equal to the angle of reflection".

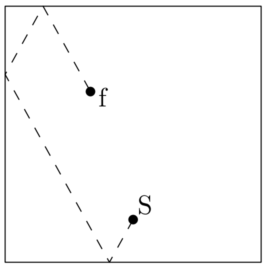


Figure 3: Saito (S) hits foe (f) after 3 bounces.

Saito knows how often he has to bounce off a wall to defeat a particular foe. He must be careful though; if his attack takes too long, his foe might be able to dodge his attack. Therefore, the distance traveled by Saito while performing his strike must be as short as possible. Can you figure out how often Saito will hit each of the four walls while performing his strike?

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- A line with three positive integer numbers L, W ($3 \leq L, W \leq 100$), and B ($0 \leq B \leq 10^5$): the length and width of the room, and the number of bounces necessary to defeat his foe.
- A line with two positive integer numbers x_S ($0 < x_S < L$) and y_S ($0 < y_S < W$): the starting coordinates of Saito.
- A line with two positive integer numbers x_f ($0 < x_f < L$) and y_f ($0 < y_f < W$): the coordinates of the foe.

The bottom left corner of the room is at $(0, 0)$. You can assume that Saito and his foe do not start at the same position. If Saito hits a corner of the room, this counts as two bounces, one for each wall. Also, Saito is able to fly over his foe while performing his strike.

[5]A ninjitsu is a technique that comes from the ninjas inner power called *Qi*.
[6]In the age of ninjas, such a foe was commonly referred to as *Boss*.

Output

For every test case in the input, the output should contain:

- One line with four integers: the number of times Saito has hit the north, east, south, and west wall, respectively. The north wall is in the positive y-direction and the east wall is in the positive x-direction. In case there are multiple possibilities, you must output all of them ordered lexicographically, each on a separate line.
- One line containing the number 0.

Examples

Input	Output
2	0 0 0 1
3 3 1	0 0 1 0
1 1	0 1 0 0
2 2	1 0 0 0
6 6 3	0
3 1	1 0 1 1
2 4	0

G Doubloon Game

Problem

Being a pirate means spending a lot of time at sea. Sometimes, when there is not much wind, days can pass by without any activity. To pass the time between chores, pirates like to play games with coins.
An old favorite of the pirates is a game for two players featuring one stack of coins. In turn, each player takes a number of coins from the stack. The number of coins that a player takes must be a power of a given integer K (1, K, K^2, etcetera). The winner is the player to take the last coin(s).
Can you help the pirates figure out how the player to move can win in a given game situation?

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with two integers S and K, satisfying $1 \leq S \leq 10^9$ and $1 \leq K \leq 100$: the size of the stack and the parameter K, respectively.

Output

For every test case in the input, the output should contain one integer on a single line: the smallest number of coins that the player to move can take in order to secure the win. If there is no winning move, the output should be 0.

Example

Input	Output
5	1
5 1	0
3 2	2
8 2	0
50 3	1
100 10	

This page is intentionally left blank.

H Walking the Plank

Problem

A bunch of pirates have successfully conquered a commercial vessel. The ship itself is too badly damaged, so the entire shipload needs to be transferred to the pirate ship.
The pirates have anchored a plank to both ships, which the pirates can use to go from one ship to the other, but it can only support one pirate at a time.
Each pirate executes the following routine:

1. Walk across the plank from the pirate ship to the commercial ship
2. Pick up an item from the cargo hold and return to the plank
3. Cross the plank back to the pirate ship with the item
4. Put the item in the cargo hold and return to the plank

Each of these four steps takes a given amount of time for each pirate. Every pirate will repeat these steps until the number of pirates on the commercial ship equals the number of items there (i.e. there's nothing more for the pirate to collect).
If a pirate gets to the plank and it's in use by another pirate, he will wait at his side of the plank. When the plank is vacated and there are pirates on both sides of the plank, the pirates on the side of the commercial vessel (carrying some item) will get to go first. At both sides of the plank, the pirates queue up, i.e. the first one to get there will be the first one to cross. Should two or more pirates arrive at the same side of the plank at the exact same time, the pirate who was slowest (i.e. the one who has taken the most time to and from the cargo hold on this ship) gets to go first.
Can you determine how long it takes before the last pirate has crossed the plank with the last item?

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with two integers N and P, satisfying $1 \leq N \leq 100,000$ and $1 \leq P \leq 1,000$: the number of items on the commercial vessel and the number of pirates, respectively.
- P lines, each with 4 integers t_1 through t_4, satisfying $1 \leq t_i \leq 1,000$: the time in seconds it takes this pirate to complete each step (as listed above).

At the start, all pirates are queued up at the plank on the pirate ship in the order given in the input (the first pirate is the first to cross over).

Output

For every test case in the input, the output should contain one integer on a single line: the time in seconds between the time the first pirate starts crossing the plank and the time the last pirate has crossed the plank carrying the last item.

Examples

Input	Output
3	63
3 3	50
10 3 10 3	24
10 3 10 3	
10 3 10 3	
3 2	
10 10 10 10	
1 9 1 8	
3 2	
3 2 3 6	
4 2 5 5	

I Parking Ships

Problem

Life on the great oceans has been good for captain Blackbeard and his fellow pirates. They have gathered so many treasures, that each of them is able to buy a house on their favorite island. The houses on this island are all placed in a long row along the beach line of the island. Next to a house, every pirate is also able to buy his own ship to do their own bit of plundering. However, this causes a whole new kind of problem.

Along the beach line there is a long pier where every pirate can park his ship. Although there is enough space along the pier for all the ships, not every pirate will be able to park in front of his house. A pirate is happy with his parking space if some part of the parking space is in front of the center of his house. Captain Blackbeard has been given the difficult task of assigning the parking spaces to the pirates. A parking space for a pirate i is a range $[a_i, b_i]$ $(a_i, b_i \in \mathbb{R})$ along the pier such that $l_i \leq b_i - a_i$, where l_i is the length of the ship of pirate i. Thus, pirate i is happy if $a_i \leq x_i \leq b_i$, where x_i is the center of the house of pirate i. Clearly, parking spaces of different pirates must be interior disjoint (the ends of ranges can coincide).

Above all, the captain wants a good parking space for himself, so he gives himself the parking space such that the center of his ship is aligned with the center of his house. Next to that, he wants to make as many pirates happy as possible. Can you help him out?

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with one integer n $(1 \leq n \leq 1,000)$: the number of pirates including the captain.
- n lines with on each line two integers x_i $(-10^9 \leq x_i \leq 10^9)$ and l_i $(1 \leq l_i \leq 10^9)$: the center of the house of the i^{th} pirate and the total length of his ship, respectively. The first pirate in the input is always the captain.

Output

For every test case in the input, the output should contain one integer on a single line: the maximum number of happy pirates using an optimal assignment of the parking spaces. This number includes the captain himself. You can assume that the space along the pier is unbounded in both directions.

Examples

Input	Output
2	5
5	3
0 6	
-5 2	
-4 1	
4 2	
5 3	
4	
0 4	
-5 4	
3 4	
5 3	

J Treasure Map

Problem

"Take 147 steps due north, turn 63 degrees clockwise, take 82 steps, ...". Most people don't realize how important accuracy is when following the directions on a pirate's treasure map. If you're even a tiny bit off at the start, you'll end up far away from the correct location at the end. Pirates therefore use very exact definitions. One step, for instance, has been defined by the 1670 Pirate Convention to be exactly two times the size of the wooden leg of Long John Silver, or 1.183 m in metric units.
Captain Borbassa was thus not at all worried when he set sail to the treasure island, having a rope with knots in it, exactly one step apart, for accurately measuring distances. Of course he also brought his good old geotriangle, once given to him by his father when he was six years old.
However, on closer inspection of the map, he got an unpleasant surprise. The map was made by the famous captain Jack Magpie, who was notorious for including little gems into his directions. In this case, there were distances listed such as $\sqrt{33}$ steps. How do you measure that accurately? Fortunately, his first mate Pythagor came to the rescue. After puzzling for a few hours, he came up with the following solution: let pirate A go 4 steps into the perpendicular direction, and hold one end of the measuring rope there. Then pirate B goes into the desired direction while letting the rope slide through his fingers, until he is exactly 7 steps away from pirate A. Pythagor worked out a formula that states that pirate B has then traveled exactly $\sqrt{33}$ steps.
Captain Borbassa was impressed, but he revealed that there were more such distances on the map. Paranoid as he is, he refuses to let Pythagor see the map, or even tell him what other distances there are on it. They are all square roots of integers, that's all he gets to know. Only on the island itself will the captain reveal the numbers, and then he expects Pyhtagor to quickly work out the smallest two integer numbers of steps that can combine to create the desired distance, using the method described above.
Pythagor knows this is not easy, so he has asked your help. Can you help him by writing a program that can determine these two integers quickly? By the way, he did ask the captain how large the numbers inside the square root could get, and the captain replied "one billion". He was probably exaggerating, but you'd better make sure the program works.
If you can successfully help the pirates, you'll get a share of the treasure. It might be gold, it might be silver, or it might even be... a treasure map!

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with one integer N, satisfying $1 \le N \le 10^9$.

Output

For every test case in the input, the output should contain two nonnegative integers, separated by a space, on a single line: the distance pirate A needs to head in the perpendicular direction, and the final distance between pirate A and B, such that pirate B has traveled $\sqrt{N}$ steps. If there are multiple solutions, give the one with the smallest numbers. If there are no solutions, the output should be "IMPOSSIBLE" (without the quotation marks) on a single line.

Examples

Input	Output
4	4 7
33	0 4
16	IMPOSSIBLE
50	50 51
101	