icpc International Collegiate Programming Contest



The 2026 ICPC North America South Central Regional Contest

Division 2

Official Problem Set

Google Cloud
Google DeepMind
icpc diamond global sponsor

JETBRAINS
icpc global sponsor programming tools

Jane Street
icpc multi-regional titanium sponsor

north america platinum sponsor

north america institutional sponsor

This page is intentionally left blank.

Problem A
What Really Is ICPC?

Time Limit: 1 second Memory Limit: 1GB

What does ICPC really stand for? You and your friend are trying to figure this out. Your friend has a suggestion of four words that they think ICPC stands for, but you're suspicious. An acronym is formed from the first letter of each word, in order. Given the four words your friend suggests, is the acronym of those four words ICPC?

Input

The input consists of a single line containing the four words your friend suggests, separated by spaces. Each word consists of at most 100 uppercase and lowercase English letters (`A`–`Z` and `a`–`z`). The first letter of each word is uppercase.

Output

Output 1 if the acronym of these words is equal to ICPC, and 0 otherwise.

Sample Input 1	Sample Output 1
International Collegiate Programming Contest	1

Sample Input 2	Sample Output 2
Introspective CoLIEgiAte PanCake CookinG	1

Sample Input 3	Sample Output 3
Undergraduate Competitive Programming Club	0

This page is intentionally left blank.

Problem B
Castle-pult

Time Limit: 1 second Memory Limit: 1GB

An alliance of three castles exists on a circle. Along the edge of this circle lie n evenly spaced zones, three of which have a castle. The zones are numbered 1 to n in order around the circle.

These castles want to be able to quickly reinforce each other with troops should one be attacked. Once an attack begins, each other castle embarks troops to move to the castle under siege. Troops walk along the edge of the circle in either direction, taking one day to move to an adjacent zone.

However, because the distances between castles may be long, each castle has decided on a more *epic* way to embark its troops. Installed in each castle is a cannon that launches its troops exactly x zones along the edge of the circle. A cannon launch takes no time. All castles use the same x.

Whenever troops set out, the cannon **always** launches them first: they choose whether it sends them x zones clockwise or counter-clockwise, and then walk the shorter way to their destination. They take the fastest such route, and they cannot use another castle's cannon.

The alliance wants to choose a positive integer for x such that the *maximum number of days* needed for troops to move between any pair of castles is *as small as possible*.

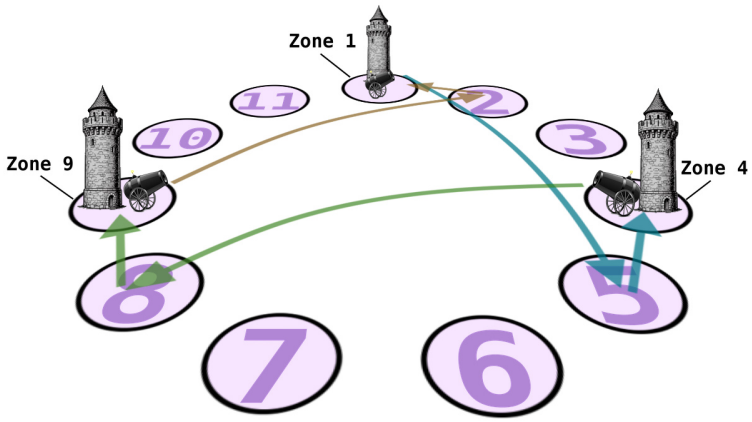


Figure B.1: Illustration of Sample 1. The alliance chooses $x = 4$. The diagram shows how troops can launch from a cannon and move at most one zone to get between any pair of castles.

Input

The input consists of a single line containing four integers: the number of zones n ($3 \leq n \leq 10^9$), followed by the positions of the three castles, each between 1 and n, inclusive. The three positions

are distinct.

Output

Output a single integer, the *smallest possible* maximum number of days needed for troops to move between any pair of castles, assuming the alliance picks x optimally.

Sample Input 1	Sample Output 1
11 1 4 9	1

Sample Input 2	Sample Output 2
12 11 1 5	2

Problem C
Crystalline Creation

Time Limit: 1 second Memory Limit: 1GB

You are given an $r \times c$ grid of tiles, where each tile is either a '`.`' or a '$\star$'. A '$\star$' represents a crystal and a '`.`' represents an empty space.

Your goal is to replicate this input by "growing" crystals on a different $r \times c$ grid that is initially empty. When you finish, your grid must match the input exactly, with crystals on all the '$\star$' tiles and none on the '`.`' tiles. There are three methods you can use to grow a crystal:

1. Simply place a crystal in any tile, with no restrictions.
2. Shine a light on a column. A crystal grows directly above the highest existing crystal in the column. If no crystal exists in this column, or the highest one is at the top edge of the grid, no new crystal grows.
3. Shine a light on a row. A crystal grows directly to the left of the leftmost existing crystal in the row. If no crystal exists in this row, or the leftmost one is at the left edge of the grid, no new crystal grows.

You may use these methods any number of times, in any order.

You prefer growing crystals rather than simply placing them on the grid, so you want to minimize the number of times you use method 1. What is the minimum number of times you must use method 1 to replicate the input, starting from an empty grid?

Input

The first line of input contains two integers r and c ($1 \leq r, c \leq 10$), the number of rows and columns of the grid.

The next r lines describe the rows of the grid from top to bottom. Each of them contains c characters, giving the tiles of its row from left to right. Every character is either '$\star$' or '`.`'.

It is guaranteed that the grid contains at least one crystal.

Output

Output a single integer, the minimum number of times method 1 must be used.

Sample Input 1	Sample Output 1
4 4 ***. .... .**. ..*.	2

Sample Input 2	Sample Output 2
3 4 **.. *... *.**	3

Problem D
Split the Bill

Time Limit: 1 second Memory Limit: 1GB

There are n friends who always eat out together and split the bill of every meal as evenly as they can. The friends only ever pay in whole dollars, so when a bill cannot be divided evenly, some of them pay more than the others. For each meal, the amounts any two friends pay differ by *at most one dollar*. For example, when 5 friends split a bill of 17 dollars, two of them pay 4 dollars and the other three pay 3 dollars. For every meal, the friends are free to decide which of them pay the extra dollars.

The friends have eaten m meals together. Given the bill of each meal, is it possible that all n friends paid the same amount in total?

Input

The first line of input contains two integers n and m ($2 \leq n \leq 1\,000$, $1 \leq m \leq 1\,000$), the number of friends and the number of meals. Each of the next m lines contains an integer between 1 and $2 \cdot 10^6$, inclusive, the bill of one of their meals in dollars.

Output

Output 1 if the friends can split the m bills so that each of them pays the same amount in total, and 0 otherwise.

Sample Input 1

```
5 3
17
3
15
```

Sample Output 1

```
1
```

Sample Input 2

```
4 2
10
7
```

Sample Output 2

```
0
```

This page is intentionally left blank.

Problem E
Painted Maze

Time Limit: 2 seconds Memory Limit: 1GB

You are given a grid with r rows and c columns. Rows are numbered 1 to r from top to bottom, columns are numbered 1 to c from left to right, and cell (i, j) is in row i and column j. Each cell is either an obstacle or painted in one of k colors. Exactly s of the k colors are special.

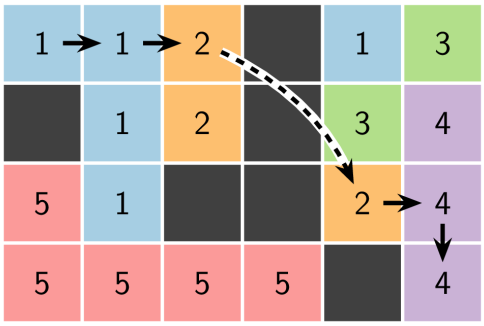


Illustration of Sample 1. The arrows show an optimal route of total cost 2. The dashed arrow is a teleport between two cells of the special color 2.

You start at the top left cell, $(1, 1)$, and wish to move to the bottom right cell, (r, c). In one move, you may step to any cell that shares a side with your current cell (you cannot move diagonally) and is not an obstacle. A step costs 0 if your current cell and the destination cell have the same color, and 1 if their colors differ. If your current cell has a special color, you may instead teleport to any other cell of that same color, at cost 0.

Find the minimum total cost to arrive at cell (r, c), or determine that it cannot be reached.

Input

The first line of input contains two positive integers r and c and two integers k and s ($2 \le r \cdot c \le 2 \cdot 10^5$, $1 \le k \le r \cdot c$, $0 \le s \le k$): the number of rows, the number of columns, the number of colors, and the number of special colors. The colors are numbered 1 to k.

The next r lines describe the rows of the grid from top to bottom. Each of them contains c integers, giving the cells of its row from left to right. Each integer is 0 if its cell is an obstacle, and the color of its cell otherwise.

Each of the next s lines contains an integer, a special color. The special colors are distinct.

It is guaranteed that neither cell $(1, 1)$ nor cell (r, c) is an obstacle.

Output

Output a single integer, the minimum total cost to arrive at cell (r, c) from cell $(1, 1)$. If cell (r, c) cannot be reached, output -1 instead.

Sample Input 1

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

Sample Output 1

```
2
```

Sample Input 2

```
3 5 3 1
1 2 0 2 1
3 1 0 1 2
3 3 0 1 1
3
```

Sample Output 2

```
-1
```

Problem F
Fork It Over

Time Limit: 1 second Memory Limit: 1GB

Anna orders lunch for the office every day. Each delivery comes with some bags of utensils, and each bag has a fork, a spoon and a knife for one person. In theory, the restaurant sends one bag for every person eating. In practice, the kitchen seems to decide the number of bags with a dice roll, so some days there are more bags than people, and some days there are fewer.

Anna hates waste. When a delivery brings more bags than there are people at lunch, she puts the extra bags in a drawer that serves as the office reserve. The drawer never fills up: it holds every bag Anna puts in it. When a delivery brings too few bags, the people left without one take a bag from the reserve, for as long as it lasts. If the reserve runs out and someone is still staring at their food without a fork, Anna walks to the store and buys exactly as many bags as are still missing. The bags she buys are used right away, so none of them end up in the reserve.

Anna has ordered lunch for n days, and the reserve was initially empty. Given the number of bags and the number of people on each day, how many bags did Anna buy at the store in total?

Input

The first line of input contains an integer n ($1 \leq n \leq 100$), the number of days. The i-th of the next n lines contains two integers between 1 and 1 000, inclusive: the number of bags in the delivery on day i, and the number of people who need a bag on day i.

Output

Output a single integer, the total number of bags Anna buys at the store over the n days.

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

This page is intentionally left blank.

Problem G
Guaranteed Drop

Time Limit: 1 second Memory Limit: 1GB

Nelson plays an online warship game, and the only way to get his favorite ship is to open containers bought from the in-game shop with coins, the game's currency. A container may give him the ship when he opens it, and players then say that the container *drops* the ship. The shop sells n types of containers, and a container of any type can drop Nelson's favorite ship. Each type has its own price p and a probability r: it costs p coins for Nelson to buy one container of that type, and the container drops the ship with probability r.

To keep unlucky players from quitting, each type of container also comes with a guarantee g. If the first $g - 1$ containers of a type that Nelson opens all fail to drop the ship, the g-th container of that type he opens is guaranteed to drop it. Each container counts only toward the guarantee of its own type, even when Nelson opens containers of other types in between. Other than the guarantee, container drops are independent of each other.

Nelson has not opened any containers yet. He buys and opens one container at a time, and before each purchase he may choose any type of container. He stops as soon as he obtains the ship. If Nelson uses a strategy that minimizes the expected number of coins he spends, how many coins does he expect to spend?

Input

The first line of input contains an integer n ($1 \leq n \leq 1\,000$), the number of types of containers. Each of the next n lines describes one type of container with an integer p ($1 \leq p \leq 10^9$), a real number r ($0 < r < 1$), and an integer g ($2 \leq g \leq 100$): the price of a container of this type in coins, the probability that it drops the ship, and the number of containers of this type that guarantees the ship. Each r is written with a single 0 before the decimal point and at least 1 and at most 4 digits after it, without trailing zeros.

Output

Output a single real number, the expected number of coins needed to get the ship with an optimal strategy. Your answer will be accepted if its absolute or relative error does not exceed 10^{-6}.

Explanation of Sample Case 1

There is only one type of container, and Nelson always pays 5 coins for the first one. With probability 0.5 it fails to drop the ship, and he pays 5 coins for a second container. With probability $0.5 \cdot 0.5 = 0.25$ both fail, and he pays 5 coins for a third container, which is guaranteed to drop the ship. In expectation he spends $5 + 0.5 \cdot 5 + 0.25 \cdot 5 = 8.75$ coins.

Sample Input 1	Sample Output 1
1 5 0.5 3	8.750000000000

Sample Input 2	Sample Output 2
4 10 0.5 100 11 0.01 2 4 0.001 5 4 0.1234 10	19.960039980004

Problem H
Office Soundproofing

Time Limit: 1 second Memory Limit: 1GB



Illustration of the panel installation in Sample 1's output. Each panel is drawn in white.

The Incessantly Chatty Programmers Company (ICPC) recently moved into a new office building. The floor is a grid of $n \times n$ square cells, and every two neighboring cells are separated by a wall. Each cell is either an office or part of the open space.

The company is soundproofing its offices. Every wall between two neighboring offices needs a soundproof panel, which can be mounted on either side of the wall, inside either of the two offices. A wall can hold at most one panel, on one side or the other. The doors and windows in the walls are no obstacle, since they are negligibly small and a panel never blocks them. Panels take up room, and it would be awkward for an office to hold more than one.

Can the company mount panels so that every wall between two neighboring offices has a panel and no office holds more than one panel? If it can, where should each panel be mounted?

Input

The first line of input contains an integer n ($2 \leq n \leq 50$), the size of the floor. The next $2n + 1$ lines draw the floor plan, with north at the top and west on the left. Each of them contains $2n + 1$ characters. On the first line of the floor plan, the third, and so on up to the last, the characters alternate between '+', a corner where walls meet, and '−', a wall running from west to east, starting and ending with '+'. On the lines between them, the characters alternate between '|', a wall running from north to south, and a cell, starting and ending with '|'. A cell is 'o' for an office or '.' for open space.

It is guaranteed that the floor has at least two offices.

Output

Output the number of panels on the first line. Then output the floor plan in the same format, showing where to mount the panels. Draw a panel on every wall between two neighboring offices,

and leave every other character unchanged. A panel is drawn as an arrow pointing into the office that holds it. On a wall running from west to east, draw '^' for a panel in the office to the north and 'v' for a panel in the office to the south. On a wall running from north to south, draw '<' for a panel in the office to the west and '>' for a panel in the office to the east.

If there are multiple ways to mount the panels, you may output any of them.

If the panels cannot be mounted to meet the requirements, output `impossible` instead.

Sample Input 1

```
2
+-+-+
|o|o|
+-+-+
|.|o|
+-+-+
```

Sample Output 1

```
2
+-+-+
|o>o|
+-+v+
|.|o|
+-+-+
```

Sample Input 2

```
4
+-+-+-+-+
|o|o|.|o|
+-+-+-+-+
|o|o|.|o|
+-+-+-+-+
|.|o|.|.|
+-+-+-+-+
|.|o|o|.|
+-+-+-+-+
```

Sample Output 2

```
8
+-+-+-+-+
|o>o|.|o|
+^+v+-+v+
|o<o|.|o|
+-+v+-+-+
|.|o|.|.|
+-+v+-+-+
|.|o>o|.|
+-+-+-+-+
```

Sample Input 3

```
3
+-+-+-+
|o|o|o|
+-+-+-+
|o|o|o|
+-+-+-+
|.|.|.|
+-+-+-+
```

Sample Output 3

```
impossible
```

Problem I
Feelin' Lucky?

Time Limit: 1 second Memory Limit: 1GB

The night before a contest, you play slot machine games on your phone. If you are lucky enough at these games, that `Math.random()` solution might just get accepted the next day!

While playing, you notice that one of the slot machines has four reels, each with some uppercase English letters on it. Every time it spins, the machine shuffles the reels into a random order, and each reel stops on one of its letters. You win if the four letters spell `ICPC` when read from left to right.

You are curious whether you can win on this machine.

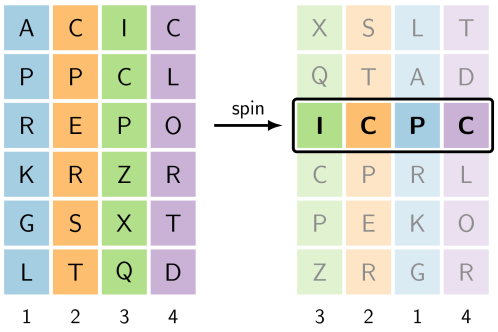


Illustration of Sample 1. This spin shuffles the reels into the order 3, 2, 1, 4 and stops them on `ICPC`.

Input

The input consists of four lines, one for each reel. Each line lists the letters on that reel: a string of distinct uppercase English letters (`A`–`Z`). All four strings have the same length, between 1 and 26, inclusive. A letter can appear on more than one reel.

Output

Output 1 if it is possible to win, and 0 otherwise.

Sample Input 1	Sample Output 1
APRKGL CPERST ICPZXQ CLORTD	1

Sample Input 2	Sample Output 2
ZCKIWB HMCRAS OLPGUD TPYEKN	0

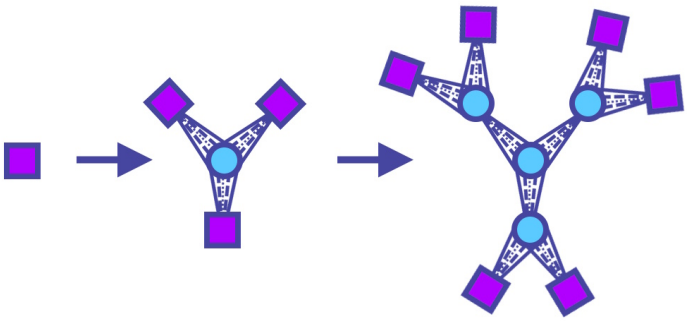
Problem J
Snow Many Ways

Time Limit: 1 second Memory Limit: 1GB

As a scientist studying how snowflakes form, you make a breakthrough in describing the nucleation process with just a few simple rules!

Specifically, a snowflake can be modeled with two types of ice crystals: frozen and unfrozen crystals. When a snowflake is born, it begins as a single unfrozen ice crystal. Then, any number of times, the snowflake may "expand". During an expansion, every unfrozen ice crystal freezes into a frozen ice crystal and splits into exactly x new unfrozen ice crystals. Here x can be any integer strictly larger than one. The value of x is the same for every unfrozen crystal within one expansion, but it does not have to be the same for every expansion.

The diagram below showcases an expansion of $x = 3$, then an expansion of $x = 2$ for a snowflake. The purple squares are unfrozen ice crystals, and the light blue circles are frozen ice crystals.



You're curious to understand all of the snowflakes that have exactly u unfrozen crystals. There are *snow many ways* to make a snowflake with u unfrozen crystals, but among all of those ways, what is the maximum number of frozen crystals that can exist on one of these snowflakes?

Input

The input consists of a single line containing an integer u ($2 \leq u \leq 10^9$), the number of unfrozen crystals on the snowflake.

Output

Output a single integer, the maximum number of frozen crystals that can exist on a snowflake with exactly u unfrozen crystals.

Sample Input 1	Sample Output 1
6	4

Problem K
2SAT Queries

Time Limit: 1 second Memory Limit: 1GB

You are playing a 2SAT guessing game with the judge. In each game, the judge has n hidden boolean variables. A 2SAT clause over two variables x and y is one of the following:

- $x \vee y$
- $\neg x \vee y$
- $x \vee \neg y$
- $\neg x \vee \neg y$

The symbol $\neg$ indicates a negation. The negation $\neg x$ is true if x is false, and false if x is true. The symbol $\vee$ indicates a logical OR. A clause is true if at least one of its two parts is true.

In one query, you pick two **different variables** and form a 2SAT clause over them, and the judge replies with its value, true or false. You win the game by guessing every hidden value correctly.

You will play at most $1\,000$ games with the judge and want to guess right in all the games using no more than $10^4 + 400$ queries. The total number of hidden variables over all games is at most 10^4.

Interaction

This is an interactive problem.

At the start of the input, you will receive a single integer between 1 and $1\,000$, inclusive, the number of games.

At the start of each game, you will receive a single integer n ($2 \le n \le 10^4$), the number of hidden variables. The variables are labeled with integers from 1 to n. The sum of n over all games is at most 10^4.

After reading n, your program should begin making queries. In each query, output a single line with two integers a and b ($1 \le |a|, |b| \le n$, $|a| \ne |b|$), the two literals of a clause. Let x_i be the value of variable i. A positive integer i represents x_i, and a negative integer $-i$ represents $\neg x_i$. For example, the query `-3 5` represents the clause $\neg x_3 \vee x_5$. You will then receive a single letter, which is `T` if the clause is true and `F` if it is false.

Once you have determined the value of every variable, output a single line with a string of n letters, whose i-th letter is `T` if x_i is true and `F` if x_i is false. This answer does not count as a query. If every value is correct and there are more games, the judge sends n for the next game, without a

separate acknowledgement. If the answer is invalid or any value is wrong, the judge will not send any further input, and you will receive a Wrong Answer verdict. After answering the last game, your program should terminate.

You may make at most $10^4 + 400$ queries in total over all games.

If a query is invalid or exceeds the query limit, you will receive the letter `X` instead of `T` or `F` in reply to that query. Upon receiving `X`, your program should terminate immediately, and you will receive a Wrong Answer verdict. The judge will not send any further input, and reading more input may cause your program to hang and receive a Time Limit Exceeded verdict.

The judge is non-adaptive: all hidden assignments are fixed before the interaction begins.

Notes

Do not forget to flush the output after printing each query and each answer. Do not output anything extra after the interaction ends.

The sample interaction only shows the format and does not represent a solving strategy.

A local testing tool, `testing_tool.py`, is provided in the attachments. See its header for usage instructions and the input file format.

Sample Interaction 1

Read	Write
2 3	
	1 3
T	
	-1 2
F	
	TFT
4	
	-4 2
T	
	1 4
F	
	-3 -2
T	
	FFTF

Problem L
Volleyball Scoring

Time Limit: 1 second Memory Limit: 1GB

In volleyball, the game consists of a series of rallies, in which the team that wins the rally scores one point and the other team's score stays the same. Whichever team scored in the previous rally always serves next, but either team is always able to score in any rally. In this problem, there is no winning score: play continues no matter how many points either team has.

The game score is kept on the scoreboard. For awareness of the score and for brevity, before each serve, only the last digit of each team's score is called out, and the team that is about to serve always has their score called out first. For example, the score of a team with 9 points serving against a team with 13 points is called out as "9 3".

Given the current scores of the two teams, determine the minimum possible number of rallies played until the score called out is "6 7".

Input

The input consists of a single line containing two integers between 0 and 100, inclusive: the score of the serving team and the score of the receiving team before the next serve.

Output

Output a single integer, the minimum possible number of rallies played until the score called out is "6 7".

Explanation of Sample Case 2

The answer is not 1 because if the receiving team wins the next rally, they also serve next, so the score is called out as "7 6".

Sample Input 1	Sample Output 1
6 17	0

Sample Input 2	Sample Output 2
6 6	11

Sample Input 3	Sample Output 3
5 7	1

Sample Input 4	Sample Output 4
7 8	8

Problem M
Mushroom Island

Time Limit: 2 seconds Memory Limit: 1GB

Many blocky adventurers often embark on long journeys looking for the rarest biome in the world: the Mushroom Island! This beautiful island is incredibly hard to find, lying alone in the middle of a vast ocean.

You are given a map of the world. On this map, every biome (excluding oceans) is marked with a *dot* at its center. Every dot is at most 100 miles away from another dot, except the one marking the Mushroom Island, which is at least $10\,000$ miles away from every other dot! All distances are Euclidean.

All of the dots are indistinguishable, so you cannot immediately tell which dot corresponds to the Mushroom Island. However, given the coordinates of each dot and the distance rules above, can you identify which dot is the Mushroom Island?

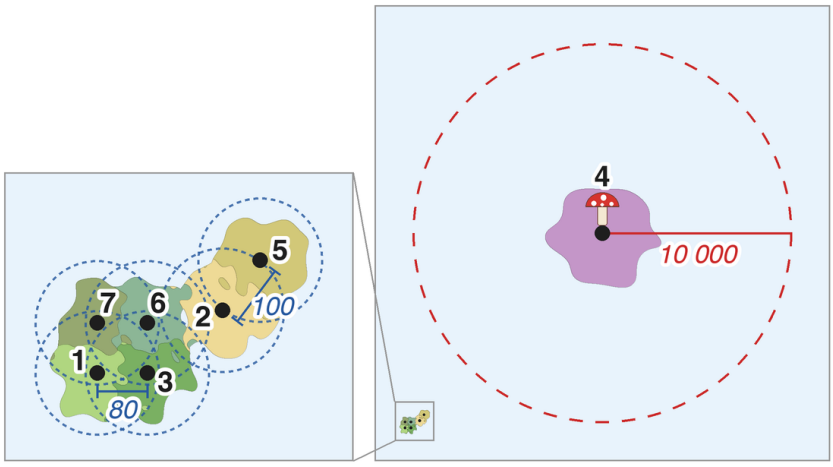


Figure M.1: Illustration of Sample 1. The dots near the origin are magnified on the left. The biome shapes are only decorative and do not matter. Every dot near the origin is at most 100 miles away from another dot, which lies on or inside its dashed circle of radius 100 miles, while dot 4, the Mushroom Island, is at least $10\,000$ miles away from every other dot.

Input

The first line of input contains an integer n ($3 \le n \le 2 \cdot 10^5$), the number of dots. Each of the next n lines contains two integers between 1 and 10^5, inclusive, the x- and y-coordinates of a dot in miles. No two dots are at the same location. The dots are numbered from 1 to n in the order they are given. It is guaranteed that the locations of the dots satisfy the conditions described above.

Output

Output a single integer, the number of the dot that marks the Mushroom Island.

Sample Input 1	Sample Output 1
7 10 10 210 110 90 10 10000 10000 270 190 90 90 10 90	4

Sample Input 2	Sample Output 2
5 100 190 150 170 49000 5960 99900 99950 99960 99910	3

This page is intentionally left blank.

2026 ICPC South Central USA Regional Division 2

Back Cover