Problem A
Astrological Sign

Harry Potter and his friends are now fourth-year students at Hogwarts School of Witchcraft and Wizardry. This year, one of their subjects is *Astrology*. To become a successful wizard, Astrology is crucial, as it allows one to predict future events or gain insights into people's personalities and relationships.

In the first lecture, Harry Potter and his friends need to learn and understand *Astrological Signs*. One's *Astrological Sign* is determined by their birthday, according to the following table:

Symbol	Astrological Sign names	Birthday
♈	Aries	March 21 – April 20
♉	Taurus	April 21 – May 20
♊	Gemini	May 21 – June 21
♋	Cancer	June 22 – July 22
♌	Leo	July 23 – August 22
♍	Virgo	August 23 – September 21
♎	Libra	September 22 – October 22
♏	Scorpio	October 23 – November 22
♐	Sagittarius	November 23 – December 21
♑	Capricorn	December 22 – January 20
♒	Aquarius	January 21 – February 19
♓	Pisces	February 20 – March 20

For example, if one's birthday is on **May 5th**, their Astrological Sign is **Taurus**, as it lies between **Apr 21st** and **May 20th**.

Today Harry Potter wants to determine the Astrological Sign of all his classmates. Please help him!

Input

The first line of the input contains a single integer t $(1 \leq t \leq 1\,000)$ — the number of Harry Potter's classmates.

In the next t line, each line contains a birthday in the format `d m`, where `d` is the date, and `m` is the first three letters of the name of the month (with the first letter in uppercase, the second and third letters in lowercase). It is guaranteed that all the given dates are valid.

Output

Print t lines, each line contains the name of the Astrological Sign.

Sample Input 1	Sample Output 1
2 5 May 30 Jul	Taurus Leo

Problem B
Battle of Hogwarts

The enemies are coming to Hogwarts School of Witchcraft and Wizardry. Please help Harry and his friends defend the school!

Hogwarts can be viewed as a grid with r rows and c columns. The rows are numbered from 1 to r from top to bottom, and the columns are numbered from 1 to c from left to right. The cell at i-th row and j-th column is denoted as (i, j).

There are 3 types of cells:

- **Wall**: The enemies cannot go into these cells.
- **Normal**: The enemies can go into these cells. Harry can use magic spell to block these cells.
- **Magic Immune**: The enemies can go into these cells. But it is not possible for Harry to block these cells.

The enemies are coming in from cell $(1, 1)$. They can move between 2 cells if they are orthogonally connected. In other words, the enemies can only move between two cells sharing a side. However, they cannot move to wall cells or normal cells blocked by Harry.

Harry must prevent the enemies from reaching cell (r, c). To do this, he can use magic spell to block some **normal** cells. Note that if either cell $(1, 1)$ or cell (r, c) is a wall, or is blocked by Harry, it means the enemies can not go from cell $(1, 1)$ to cell (r, c).

However, time is running out! Harry needs to know what is the minimum number of **normal** cells he need to block. Please help him!

Input

The input contains multiple test cases. Each test case consists of:

- The first line contains two positive integers r and c ($1 \le r \cdot c \le 10^6$).
- In the next r lines, the i-th line contains exactly c characters. Each character can be one of the following:
  - `#` representing a **wall**.
  - `.` representing a **normal** cell.
  - `@` representing a **magic immune** cell.

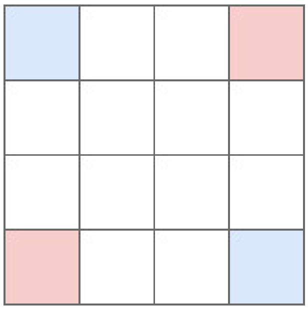
The input terminates with two 0, and you don't have to process this case.

The sum of $r \cdot c$ in all test cases does not exceed 10^6.

Output

For each test case, print exactly a single line containing the minimum number of **normal** cells Harry needs to block to prevent the enemies from reaching the cell (r, c). If it is not possible, print -1.

Explanation of the first sample



In the above figure, **walls** are red, **normal** cells are white and **magic immune** cells are blue.

Harry can block 2 cells $(2, 3)$ and $(3, 2)$. It is not possible for Harry to block 1 or less cell and prevent enemies from reaching cell $(4, 4)$.

Sample Input 1

```
4 4
@..#
....
....
#..@
4 4
@..#
..#.
.#..
#..@
0 0
```

Sample Output 1

```
2
0
```

Problem C
Cable Car

At 3 147.3 meters high, Fansipan is the tallest mountain in the Indochina peninsula. To promote tourism, n stations were built on the mountain, numbered from 1 to n.

Two companies, Mobi and Vina are in charge of operating cable cars connecting the stations. Each of the two companies have k cable cars. The i-th cable car of Mobi connects two stations MS_i and ME_i. The i-th cable car of Vina connects two stations VS_i and VE_i.

Two stations are called connected by a company, iff we can go from one station to the other by using cable cars only of that company. To achieve peaceful cooperation, the two companies agreed with the following conditions:

- For every valid i, $MS_i < ME_i$ and $VS_i < VE_i$.
- All MS_i are unique, all ME_i are unique.
- All VS_i are unique, all VE_i are unique.
- For any $i \neq j$, if $MS_i < MS_j$, then $ME_i < ME_j$.
- For any $i \neq j$, if $VS_i < VS_j$, then $VE_i < VE_j$.
- No pair of stations is connected by both companies. In other words, for every pair of stations i and j, if i and j are connected by Mobi, they should not be connected by Vina, and vice versa.

Given n and k, your task is to check whether it is possible for Mobi and Vina to each operates k cable cars, satisfying all the above conditions.

Input

The input contains two integers n and k, separated by a single space $(1 \leq k < n \leq 100)$.

Output

For each test case, if it is not possible to satisfy all the conditions, print 'NO'. Otherwise, print 'YES', followed by $2 \cdot k$ lines. In the first k lines, the i-th line contains two integers MS_i and ME_i. In the last k lines, the i-th line contains two integers VS_i and VE_i.

Sample Input 1	Sample Output 1
3 1	YES 1 2 1 3

Sample Input 2	Sample Output 2
3 2	NO

Problem D
Dividing Kingdom

The Kingdom of Byteland has n cities, numbered from 1 to n. There are exactly $n-1$ roads in the kingdom, each connects a pair of cities. Using these roads, it is possible to go from any city to any other city.

The king of Byteland wants to divide the kingdom into two halves and pass down each half to one of his two sons. This would be done by removing exactly one road, splitting the kingdom into two connected components.

This turns out to be a non-trivial task! To avoid fighting between his sons, the king needs to divide the kingdom fairly. For each half, he defines its **diameter** as the longest simple path connecting two cities in that half. The king wants the difference between the diameters of the two halves to be minimum.

Note: A simple path from city s to city t is an ordered sequence of cities $v_0 \to v_1 \to v_2 \to \ldots \to v_k$, where $v_0 = s, v_k = t$, and all v_i are unique. For each valid index i, v_i and v_{i+1} are connected directly by some road. The length of a path is the sum of the length of all roads connecting v_i and v_{i+1}.

Please help the king!

Input

The input contains multiple test cases, each test case is presented as below:

- The first line contains a positive integer n $(2 \le n \le 3 \cdot 10^5)$ — the number of cities. The sum of n in all test cases does not exceed 10^6.
- In the next $n-1$ lines, the i-th line $(i = 1 \ldots n-1)$ contains two integers p_i and l_i $(1 \le p_i \le i, 1 \le l_i \le 10^9)$, indicating that two cities p_i and $i+1$ are connected by a road of length l_i.

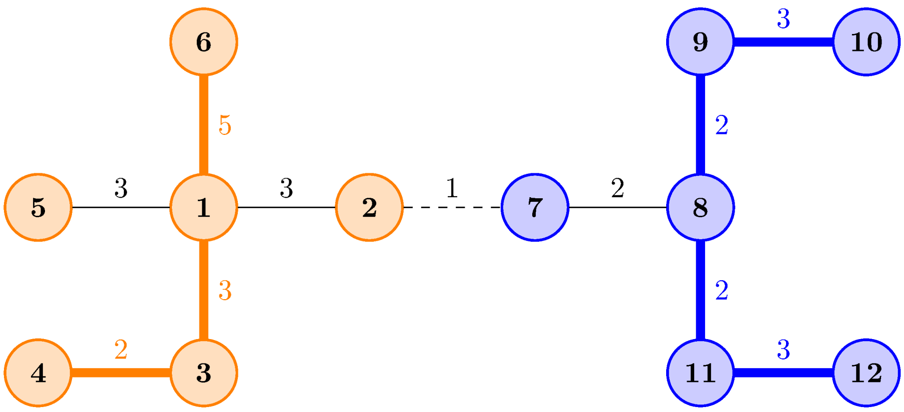
The input ends with a line containing a single 0 which is not a test case.

Output

For each test case, print a single line containing a single integer — the minimum difference between the two diameters.

Explanation of the first sample

The figure below demonstrates the first test case. One way to obtain an optimal solution is to remove the road between city 2 and city 7 (marked by dashed line). By removing this road, the kingdom is split into two halves, whose cities are marked in orange and blue colors. The longest simple paths are marked in bolder colors.



Sample Input 1

```
12
1 3
1 3
3 2
1 3
1 5
2 1
7 2
8 2
9 3
8 2
11 3
4
1 10
2 10
3 10
4
1 20
2 10
3 10
0
```

Sample Output 1

```
0
0
10
```

Problem E
Easy Arithmetic

Harry is now a third-year student at Hogwarts School of Witchcraft and Wizardry. Contrary to popular belief, arithmetic (known as Arithmancy in the wizard world) is crucial to becoming a powerful wizard.

For today's Arithmancy homework, Harry is given a super long expression s of length n, containing digits, the + and − operators. The i-th $(1 \le i \le n)$ character of s is denoted by s_i.

Harry must answer q queries of the following two types:

1. $?\ \ell\ r$: Compute the value of the expression from the ℓ-th character to the r-th character, inclusive. As the value of this expression can be very large, Harry must calculate it modulo $998\,244\,353$.

2. $!\ i\ c$: Change s_i to the character c.

Please help Harry!

Input

- The first line contains n characters representing the expression s $(1 \le n \le 10^5)$. It is guaranteed that all characters are either digits or the + and − operators, and no two consecutive characters in s are operators.

- The second line contains q $(1 \le q \le 10^5)$.

- In the next q lines, each line contains one query:
  - $?\ \ell\ r$ $(1 \le \ell \le r \le n)$: representing a query of the first type. It is guaranteed that s_r is a digit.
  - $!\ i\ c$ $(1 \le i \le n)$, c is either a digit or + or −: representing a query of the second type. It is guaranteed that after this query, no two consecutive characters are operators.

Output

For each query of the first type, print the value of the expression, modulo $998\,244\,353$.

Sample Input 1	Sample Output 1
123-456+789 10 ? 2 5 ? 7 9 ? 4 6 ! 1 + ! 4 0 ? 1 7 ! 1 - ? 1 3 ? 4 7 ? 4 5	19 13 998244308 230456 998244330 456 4

Problem F
Fluffy Cat

You are working on your new mobile game *Fluffy Cat*. With simple yet engaging game play, it will surely become a top mobile game!

In this game, a single player controls a cat in an infinite grid. There is a mouse hidden somewhere in this grid. To win the game, the player must catch the mouse. The game consists of several rounds. In each round:

- The player moves the cat at most two (possibly zero) steps. In each step, the cat can move one unit of length towards any of the following directions: up, down, left and right. More precisely, in one step, the cat can move from the current point (x, y) to any of these 4 points: $(x - 1, y)$, $(x + 1, y)$, $(x, y - 1)$ and $(x, y + 1)$.
- If the cat and the mouse are at the same point, the mouse must not move, the game ends immediately and the player wins.
- Otherwise, the mouse moves exactly one unit of length towards any of the directions up, down, left and right. The mouse moves using a pre-written algorithm which is hidden from the player.
- If the cat and the mouse are at the same point, the game ends immediately and the player wins. Otherwise, the game continues to the next round.

At the end of each round, the player is informed the squared Euclidean distance between the cat and the mouse. If this distance equals to 0, the game has ended and the player has already won. If the cat is at point (x_c, y_c) and the mouse is at point (x_m, y_m), then the squared Euclidean distance between the cat and the mouse is defined as: $(x_c - x_m)^2 + (y_c - y_m)^2$.

If the player fails to catch the mouse within $10\,000$ rounds, he loses the game. Your task is to win the game no matter how the mouse plays.

Interaction

In each round:

- First, your program writes an integer k $(0 \leq k \leq 2)$ — the number of steps you want the cat to move, followed by a space and k characters, each must be one of 'LRUD', representing the direction you want the cat to move.
- Your program then reads an integer d — the squared Euclidean distance between the cat and the mouse. If $d = 0$, you win and your program should terminate immediately.

Initially, the squared Euclidean distance between the cat and the mouse is at most 10^6. The cat must catch the mouse within $10\,000$ rounds.

Communication example

In this example, the cat is initially at $(0, 0)$, the mouse is at $(1, 0)$. The mouse is programmed to always move right (from (x, y) to $(x+1, y)$). Note that your program knows neither the initial positions nor the strategy of the mouse. Also, this strategy is not used in our interactor.

Your output (standard output)	Kattis' answer (standard input)	Interpretation
2 `RL`		The cat moves right to $(1, 0)$ and then left to $(0, 0)$.
	4	The mouse moves right to $(2, 0)$, and the squared distance between the cat and the mouse is now 4.
2 `RU`		The cat moves right and then up to $(1, 1)$.
	5	The mouse moves right to $(3, 0)$, and the squared distance between the cat and the mouse is now 5.
2 `RR`		The cat moves right twice to $(3, 1)$.
	2	The mouse moves right to $(4, 0)$, and the squared distance between the cat and the mouse is now 2.
2 `RD`		The cat moves to $(4, 0)$. The cat and the mouse are now at same position. The player wins.
	0	Your program should terminate immediately after reading 0.

Notes

In this problem, the interactor is adaptive — the jury program can move the mouse using different strategies, depending on your program's output.

When you write the solution for the interactive problem it is important to keep in mind that if you output some data it is possible that this data is first placed to some internal buffer and may be not directly transferred to the interactor. In order to avoid such situation **you have to use special 'flush' operation each time you output some data**. There are these 'flush' operations in standard libraries of almost all languages. For example, in C++ you may use `fflush(stdout)` or `cout « flush` (it depends on what do you use for output data — `scanf/printf` or `cout`). In Java you can use method 'flush' for output stream, for example, `System.out.flush()`. In Python you can use `stdout.flush()`.

Problem G
Greatest Pair

You are given a tree with n vertices. Each edge has a **weight**, and each vertex has a **label**. We denote the label of vertex i as $label(i)$.

A simple path from vertex s to vertex t is defined as an ordered sequence of vertices $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$, where $v_0 = s, v_k = t$, and all v_i are unique. For each valid index i, v_i and v_{i+1} are connected directly by an edge. Note that there exists a simple path between every pair of vertices in a tree.

We define:

- $dist(u, v)$ as the sum of the **weight** of all edges on the simple path from u to v.
- $greatness(u, v) = dist(u, v) \cdot gcd(label(u), label(v))$.

Please find the two different vertices u and v with maximum $greatness(u, v)$.

Input

The input contains multiple test cases, each test case is presented as below:

- The first line contains a single integer n ($2 \leq n \leq 10^5$). The sum of n among all test cases does not exceed 10^5.
- The second line contains n integers, the i-th integer is $label(i)$ ($1 \leq label(i) \leq 5 \cdot 10^5$).
- In the next $n - 1$ lines, each line contains three integers u, v and w ($1 \leq u, v \leq n, 1 \leq w \leq 10^6$) describing an edge of weight w connecting two vertices u and v.

The input ends with a line containing a single 0 which is not a test case.

Output

For each test case, print a single line containing the maximum value of $greatness(u, v)$.

Sample Input 1	Sample Output 1
2 10 10 1 2 10 0	100

Problem H
Highway to Mount Fansipan

In the year 3 030, *Highway to mount Fansipan* is the most popular TV game show in Vietnam.

In this game show, four contestants are given a crossword puzzle. The crossword consists of n horizontal words. The first letters of the n words form a **hidden message** when read from top to bottom. The first contestant who finds the **hidden message** wins the game.

I	N	V	I	T	E
C	A	T			
P	A	R	T	Y	
C	O	W			

In the above figure, the crossword has 4 horizontal words: *invite*, *cat*, *party* and *cow*. The first letters of these 4 words form the hidden message *icpc*.

This year, you are tasked with creating a crossword for *Highway to mount Fansipan*. You are given the following requirements:

- There must be n horizontal words. The i-th word must have exactly w_i characters.
- You are given a dictionary with d words. Each horizontal word and the hidden message must be in the given dictionary.
- The n horizontal words and the hidden message must be pairwise different.

As an expert programmer, generating a crossword satisfying the above requirements is an easy task for you. But you are wondering how many different ways of generating crossword there are. Two crosswords are considered different if at least one of the n horizontal words are different.

Input

The first line of the input contains a single integer T — the number of test cases. T test cases follow, each test case is presented as below:

- The first line contains a positive integer n — the number of horizontal words. The sum of n in all test cases does not exceed 10^5.
- The second line contains exactly n positive integers, $w_1, w_2, \ldots, w_n$ $(1 \leq w_i \leq 50)$, where w_i is the length of the i-th horizontal word.
- The third line contains a positive integer d. The sum of d in all test cases does not exceed 10^5.

- In the next d lines, each line contains a single word in the dictionary. Each word contains between 1 and 50 lowercase English letters. No two words are equal. The total length of all words in all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output a single integer — the number of different ways to generate the crossword, modulo $10^9 + 7$.

Sample Input 1

```
2
4
6 3 5 3
5
invite
icpc
party
cat
cow
4
6 3 5 3
4
invite
party
cat
cow
```

Sample Output 1

```
2
0
```

Problem I
Infinite 2D Array

We define an infinite two dimensional array F using the following formula:

- $F_{0,0} = 0$,
- $F_{0,1} = F_{1,0} = 1$,
- For $i \geq 2$, $F_{i,0} = F_{i-1,0} + F_{i-2,0}$,
- For $i \geq 2$, $F_{0,i} = F_{0,i-1} + F_{0,i-2}$,
- For $i, j \geq 1$, $F_{i,j} = F_{i-1,j} + F_{i,j-1}$.

Here are the first few values of F:

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	1	1	2	3	5	8
1	1	2	3	5	8	13	21
2	1	3	6	11	19	32	53
3	2	5	11	22	41	73	126
4	3	8	19	41	82	155	281
5	5	13	32	73	155	310	591
6	8	21	53	126	281	591	1182

Given x and y, your task is to calculate $F_{x,y}$.

Input

The input consists of only a single line, containing 2 integers x and y $(0 < x, y < 10^6)$.

Output

Print a single integer — the value of $F_{x,y}$, modulo $10^9 + 7$.

Sample Input 1	Sample Output 1
`2 2`	`6`

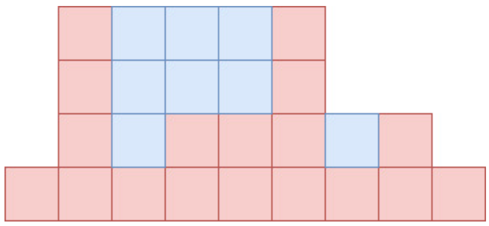
Sample Input 2	Sample Output 2
`1 5`	`13`

Problem J
Just Enough Water

As you may remember, in The 2 016 ICPC Nha Trang Regional Contest, the problem *Reservoir*, we have built a reservoir near the Red River. The reservoir can be viewed as a rectangular box with unit-length width. Its length can be divided into n sections of unit-length, the i-th section having height of h_i units. We assume that the left and right of the reservoir has height of 0, i.e. $h_0 = h_{n+1} = 0$.

After the rain, some of the sections of the reservoir are filled with water. Naturally, water flows from higher places to lower places, so water flows to both the left and the right of the reservoir.

An example of the cross section of the reservoir along its length and height dimensions is shown in the following illustration:



In the above picture:

- $n = 9, h = [1, 4, 1, 2, 2, 4, 1, 2, 1]$.
- The reservoir is filled with 8 units of water.

It was found that the reservoir does not hold enough water, thus we have decided to raise the height of some sections. It costs 1 dollar to raise the height of one section by 1 unit. We have a total budget of k dollars.

What is the maximum number of units of water the reservoir can hold?

Input

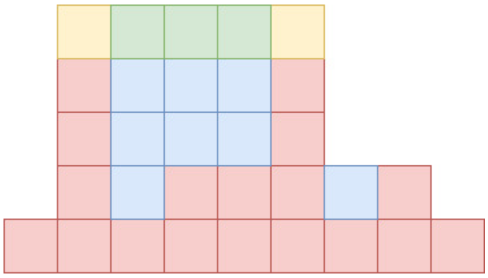
- The first line contains two integers n and k $(1 \le n, k \le 12)$.
- The second line contains exactly n integers $h_1, h_2, \ldots, h_n$ $(1 \le h_n \le 10^9)$, representing the height of n sections.

Output

Print a single integer — the maximum amount of water the reservoir can hold.

Explanation of the first sample

Initially, the reservoir looks like the above-mentioned figure. The figure below demonstrates an optimal way to maximize the amount of water that the reservoir can hold. Yellow cells show how we raise sections' height. Green cells show extra water that the reservoir can hold.



Sample Input 1	Sample Output 1
9 2 1 4 1 2 2 4 1 2 1	11

Problem K
Kingdom of Cats

You probably still remember the problem *Kingdom of Kittens* from The 2018 ICPC Asia Hanoi Regional Contest. It has been two years since then, the kingdom has since evolved into *Kingdom of Cats*, with even more challenging problems.

As their kingdom grows, the cats need to set a new border for their kingdom. They have found n beautiful bushes. They decided to use these bushes to mark the border of their kingdom.

Unfortunately, cats are not so good at geometry, so they want the border to be a polygon with exactly four sides. Additionally, to avoid confusion, for each pair of points inside the polygon, all points on the segment connecting the pair must also lie inside the polygon. In other words, the border must be a convex polygon. As their plan is still under discussion, the cats want you to tell them how many ways to choose the border satisfying all their needs.

More formally, given n points on a Cartesian coordinate plane, you should count the number of convex polygons with positive area whose all 4 vertices are amongst the n points.

Input

The input contains multiple test cases. Each test case is descried as below:

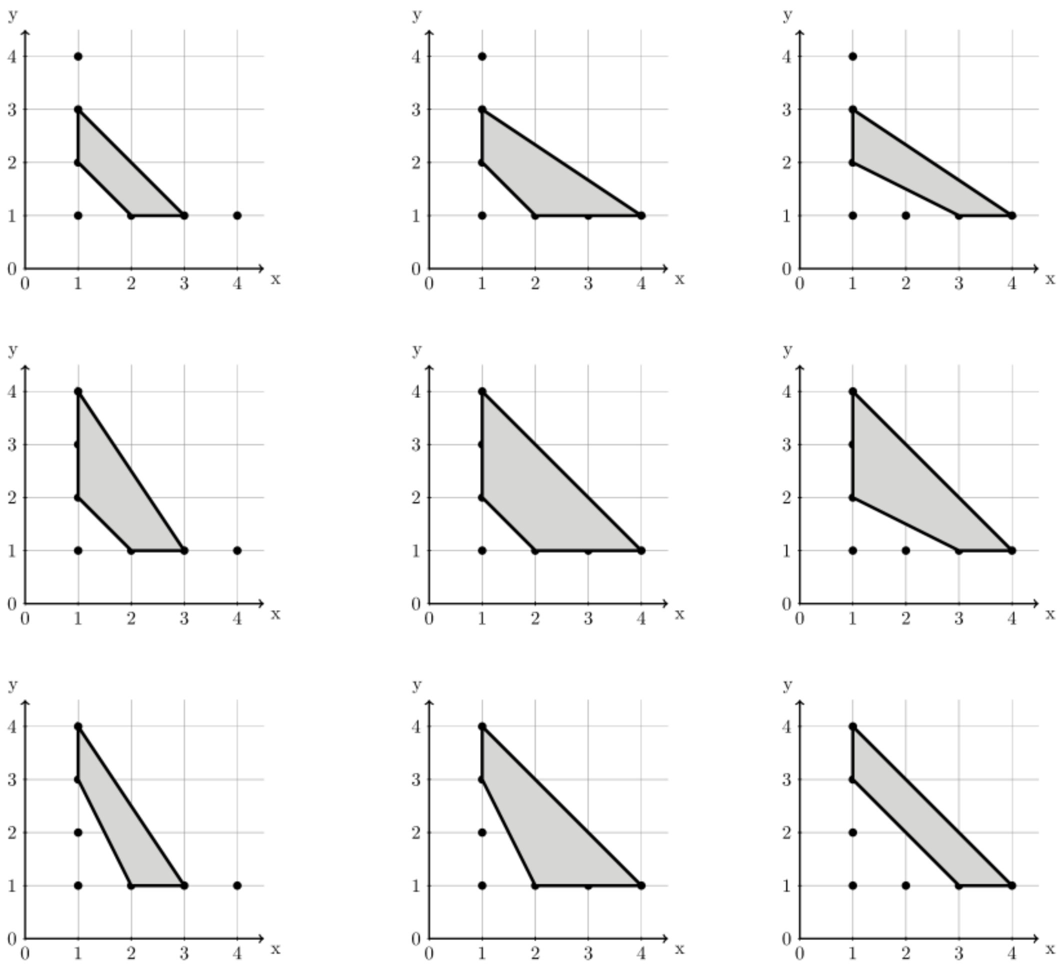
- The first line contains a single positive integer n — the number of points $(1 \le n \le 50)$. The sum of n in all test cases does not exceed 500.
- In the next n lines, the i-th line contains two integers x_i and y_i — the coordinates of the i-th point $(-10^9 \le x_i, y_i \le 10^9)$. It is guaranteed that no two points have the same coordinates.

The input is terminated with a line containing a single 0.

Output

For each test case, print a single line containing the number of different convex polygons satisfying the given conditions.

Explanation of the first sample



Sample Input 1

```
7
4 1
3 1
2 1
1 1
1 2
1 3
1 4
4
0 0
0 2
2 0
2 2
0
```

Sample Output 1

```
9
1
```

Problem L
Looping Around

Chubby Trung and skinny Hanh are best friend. They are smart and lovely kids, and they are both very good at math and programming. One day, chubby Trung set a puzzle to challenge skinny Hanh. Trung gives a list of n distinct points on a Cartesian coordinate plane to Hanh and asks Hanh to draw a single loop connecting all these points with some conditions:

- The loop consists of exactly n segments that are parallel to the axes.
- Each segment's two ends must be 2 of the given points. Other than these 2 points, the segment must not go through any other points from the list.
- Two consecutive segments in the loop must be perpendicular (i.e. they must form a 90 degree angle), and they must have exactly one intersection (which is at their common end).
- The loop must go through all n points. The first and last point of the loop must be the same.
- The loop must not self-intersect.

In the following examples, the first figure shows a valid loop. The second figure shows an invalid loop because there are segments' ends at $(2, 2)$ which is not in the list. The third one is also invalid because it does not go through all n points. And the last figure is also invalid because there exists 2 consecutive segments that are not perpendicular.

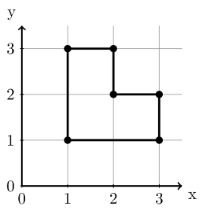


Fig. 1

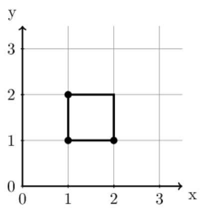


Fig. 2

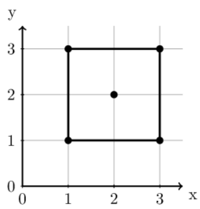


Fig. 3

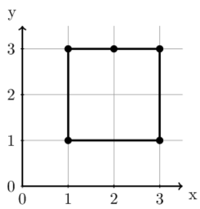


Fig. 4

Your task is to help skinny Hanh determine whether it is possible to create such loop.

Input

The input starts with an positive integer t – the number of test cases. Then t test cases follow, each has the following format:

- The first line consists of an integer n – the number of points in the list ($1 \le n \le 2 \cdot 10^5$). The sum of n in all test cases does not exceed 10^6.
- The i-th line of the next n lines contains 2 integers x_i, y_i describing the i-th point ($0 \le |x_i|, |y_i| \le 10^9$). It is guaranteed that no two points have the same coordinates.

Output

For each test case, if it is possible to draw the loop, print ‘YES’; otherwise, print ‘NO’.

Sample Input 1

```
2
6
1 1
1 3
2 2
2 3
3 1
3 2
3
1 1
1 2
2 1
```

Sample Output 1

```
YES
NO
```

Sample Input 2

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

Sample Output 2

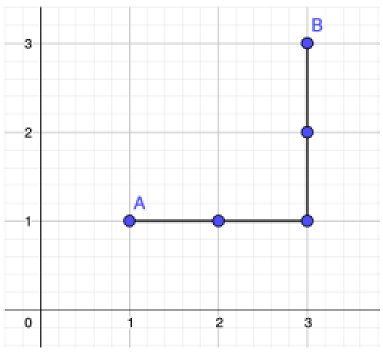
```
NO
NO
```

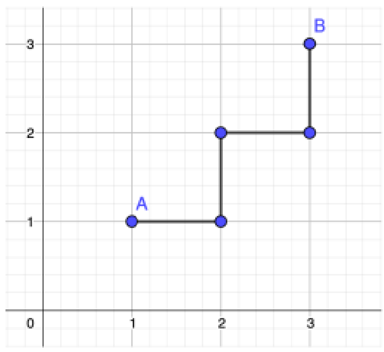
Problem M
Malfunctioning Robot

Arthur is participating in the Vietnamese Robotic Olympiad $2\,020$. In the first round, candidates must build a path-finding robot. More precisely, the organizers have prepared an infinite, obstacle-free grid. One candidate's robot is placed at a starting position (x_1, y_1) and must find a path to a target position (x_2, y_2).

Arthur's robot is quite simple: it can only move in four directions: up, down, left and right, i.e. from (x, y), it can moves to $(x, y+1)$, $(x, y-1)$, $(x-1, y)$ and $(x+1, y)$.

The first round is starting soon. However, Arthur just realized that some of the robot's components are broken, and thus the robot cannot move in the same direction in two consecutive moves (i.e. it always changes direction after each move). For example, if the robot needs to go from $(1, 1)$ to $(3, 3)$, the robot can use the path shown on the right, but not the path shown on the left.





Please help Arthur find the minimum number of moves for his robot to get from some starting position to some ending one.

Input

The input starts with T - the number of test cases ($1 \le T \le 1\,000$), then T test cases follow. Each test case is presented in a single line with 4 integers x_1, y_1, x_2 and y_2 which denote the starting and target position of the robot.

The absolute value of all integers in the input do not exceed 10^9.

Output

For each test case, print the minimum number of moves for Arthur's robot to get from (x_1, y_1) to (x_2, y_2)

Sample Input 1	Sample Output 1
2 1 1 3 3 0 1 1 1	4 1