Sylhet ICPC 2024 Collaborative Challenge: Episode 1

Problem Set

CoderOJ

A. Free Food

Time: 1 s
Memory: 128 MB

Do you know what attracts almost any university student to participate in an event? Yes, free food. It doesn't matter whether the event involves a long (sometimes boring) seminar. As long as free food is served for the event, students will surely come.
Suppose there are N events to be held this year. The i-th event is scheduled from day s_i to day t_i, and free food is served for that event every day from day s_i to day t_i (inclusive). Your task in this problem is to find out how many days are there in which free food is served by at least one event.
For example, let's assume there are $N = 3$ events. The first event is held from day 10 to 14, the second event is held from day 13 to 17, and the third event is held from day 25 to 26. The days in which free food is served by at least one event are 10,11,12,13,14,15,16,17, 25, 26 - for a total of 10 days. Note that multiple events serve free food on days 13 and 14.

Input

The input starts with an integer T - the number of test cases you need to solve.
Each test case starts with an integer $N(1 \leq N \leq 100)$ denoting the number of events. Each of the next N lines contains two integers s_i and t_i $(1 \leq s_i \leq t_i \leq 365)$ denoting that the *i-th* event will be held from s_i to t_i (inclusive), and free food is served for all of those days.

Output

For each test case, print an integer denoting the number of days in which free food is served by at least one event.

Examples

Input	Output
3 3 10 14 13 17 25 26 2 1 365 20 28 4 29 29 48 48 102 102 94 94	10 365 4

B. Dancing Digits

Time: 5 s
Memory: 128 MB

Digits like to dance. One day, 1, 2, 3, 4, 5, 6, 7, and 8 stand in a line to have a wonderful party. Each time, a male digit can ask a female digit to dance with him, or a female digit can ask a male digit to dance with her, as long as their sum is prime. Before every dance, exactly one digit goes to who he/she wants to dance with - either to its immediate left or immediate right. For simplicity, we denote a male digit x by itself x, and denote a female digit x by −x. Suppose the digits are in order {1, 2, 4, 5, 6, -7, -3, 8}. If -3 wants to dance with 4, she must go either to 4's left, resulting {1, 2, -3, 4, 5, 6, -7, 8} or his right, resulting {1, 2, 4, -3, 5, 6, -7, 8}. Note that -3 cannot dance with 5, since their sum 3+5=8 is not a prime; 2 cannot dance with 5, since they're both male.
Given the initial ordering of the digits, find the minimal number of dances needed for them to sort in increasing order (ignoring signs of course).

Input

The input consists of at most 20 test cases. Each case contains exactly 8 integers in a single line.
The absolute values of these integers form a permutation of {1, 2, 3, 4, 5, 6, 7, 8}. The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and the minimal number of dances needed. If they can never be sorted in increasing order, print '-1'.

Examples

Input	Output
1 2 4 5 6 -7 -3 8 1 2 3 4 5 6 7 8 1 2 3 5 -4 6 7 8 1 2 3 5 4 6 7 8 2 -8 -4 5 6 7 3 -1 0	Case 1: 1 Case 2: 0 Case 3: 1 Case 4: -1 Case 5: 3

C. Undo History

Time: 1 s
Memory: 128 MB

Robin is using a peculiar text editor to write a sequence of lines of text. The editor consists of three parts: a **results window**, a **text buffer**, and an **undo history**. More details about the three parts follow.

- The results window contains a sequence of strings: the lines of text you already wrote. Initially, the results window is empty.
- The text buffer contains a string: the line you are writing at the moment. Initially, the string in the text buffer is empty.
- The undo history contains a sequence of strings: all the past states of the text buffer. Initially, the undo history contains a single element: an empty string.

You are given a sequence of lines. Robin would like to print the contents of these lines into the results window (At the end, the sequence of strings stored in the results window must be precisely equal to the lines. Order of elements matters.). Additionally, Robin would like to do so as quickly as possible. He is able to take the following actions:

- Robin may type a lowercase letter. The letter is appended to the text buffer. The new text buffer is then added as a new element of the undo history. (For example, if the text buffer currently contains "do", then pressing 'g' changes the text buffer to "dog" and then stores "dog" into the undo history.)
- Robin may press Enter. When he does so, the current content of the text buffer is printed to the results window as a new line, after the lines that were printed earlier. The text buffer remains unmodified. (For example, if the text buffer contains "dog" and Robin presses Enter, "dog" will be appended to the results window, and the text buffer still contains "dog".)
- Robin may use two mouse clicks to restore any entry from the undo history to the text buffer. This operation does not modify the undo history.

Calculate the minimum total number of button presses (keyboard and mouse) that Robin needs to print all the given lines into the results window.

Input

Input starts with a positive integer **T**, denoting the number of test cases ($T < 200$). Each test case starts with an integer **n**, the number of strings in that case ($1 \leq n \leq 100$). This line is followed by **n** lines, each containing a string of no more than 50 characters. Each element of a string will contain only lowercase letters ('a' – 'z').

Output

Print the minimum number of operations required for the given task. See the sample outputs for the exact format of the output.

Examples

Input	Output
5 2 tomorrow today 2 a b 4 a ab abac abacus 3 absolutely abs absolute	Case #1: 15 Case #2: 6 Case #3: 10 Case #4: 17 Case #5: 39

Input	Output
5 pyramid sphinx sphere python serpent	

Notes

Case 1:

- Type 't'. The text buffer now contains "t", and the undo history now contains "" and "t".
- Type 'o'. The text buffer now contains "to", and the undo history now contains "", "t", and "to".
- Using six more keypresses, type the letters in "morrow". The text buffer now contains "tomorrow" and the undo history contains all prefixes of "tomorrow". The results window is still empty.
- Press Enter. The results window now contains one string: "tomorrow".
- Click the mouse twice to restore "to" from undo history.
- Using another three keypresses, type the letters in "day".
- Press Enter. The results window now contains "tomorrow" and "today", in this order, and we are done.

The total number of button presses was 8 (typing "tomorrow") + 1 (Enter) + 2 (mouse) + 3 (typing "day") + 1 (Enter) = 15.

Case 2:
After typing "a" and pressing enter, we need to restore the empty string (which is always present at the top of the undo buffer) before typing "b".

Case 3:
There are times when it is not necessary to use the undo history at all.

D. Tic-Tac-Toe

Time: 1 s
Memory: 128 MB

Alice and Bob are playing a game of tic-tac-toe, to be precise a variant of tic-tac-toe called Notakto. In a game of Notakto, there will always be a winner.
Notakto is played on 3×3 tic-tac-toe board with both players playing X. A game ends when the board contains three consecutive X, at which point the player to have made the last move loses the game.
Given a configuration of the board, you have to find out the winner. Alice always makes the first move.

Input

You are given an integer **n** ($0 < n < 200$), number of games.
For each game, you are given a 3×3 grid, each grid cell is empty (denoted by a dot), or occupied by a cross (denoted by X).

Output

For each game, print the winner of that game, in the format "Game #x: winner". Here, x is the game number.

Examples

Input	Output
1 .X. .X. ..X	Game #1: Bob

Input	Output
2 .X. .XX X.X ... .XX ...	Game #1: Bob Game #2: Alice

E. Reciting to the Caterpillar

Time: 1 s
Memory: 128 MB

What a strange day it has been! Alice has just escaped from a most unusual situation, where she found herself trapped inside the White Rabbit's house after a new encounter with a "Drink Me" bottle, and growing so much that her body barely fitted inside the house.
Anyway, she transformed back to small proportions after some rocks that were thrown at her became little cakes that she promptly ate. A wise decision at the moment, but now that she's outside the house, walking through a forest, she misses her normal size. Alice was pondering all of this when suddenly she came across a Caterpillar quietly smoking a long hookah on top of a mushroom.

Alice meets the Caterpillar

Alice explained her situation to the Caterpillar, and how she thinks that she's changed somehow. "I can't remember things as I used--and I don't keep the same size for ten minutes together!" she says.
"Can't remember what things?" said the Caterpillar.
"Well, I've tried to say *How Doth the Little Busy Bee*, but it all came different!" Alice replied in a very melancholy voice.
"Repeat, *You are Old, Father William*", said the Caterpillar. Alice folded her hands and began:
`You are old, Father William,' the young man said,
And your hair has become very white;
And yet you incessantly stand on your head--
Do you think, at your age, it is right?' ...
Alice goes on, worrying that the words are coming out all wrong, while the Caterpillar patiently listens, paying attention to the form of her verses, particularly the rhyming schemes. The first verse that Alice declaims, for example, has a rhyme scheme *ABAB*.
Do you think you can figure out how many rhyme schemes can be used in a verse of *N* lines? For example, if *N*=3, then there are 5 possible schemes: *AAA*, *AAB*, *ABA*, *ABB* and *ABC*.

Input

The first line of the input contains a positive integer **T**, that denotes the number of test cases ($T \leq 50$).
Each test case comes in a single line, and contains a single integer **N**, which is the number of lines of a verse ($1 \leq N \leq 500$).

Output

For each test case, print the number of the test case, and the number of rhyme schemes that could be used in a verse of *N* lines. Since this number can be very large, print the result modulo 1000000007.

Examples

Input	Output
3 3 11 101	Case 1: 5 Case 2: 678570 Case 3: 513369106

F. Lighting Strategy

Time: 1 s
Memory: 128 MB

Ronju, the night guard at the headquarters of **Lavish Office Buildings Ltd.**, is responsible for lighting up a large grass field in front of the building each evening. However, due to the extensive size of the field and the numerous lights installed, it becomes exhausting for him to walk to each individual light to turn it on.To simplify his task, he has devised a clever plan: he will replace the traditional switches with light-sensitive triggers. A local electronic store sells these innovative trigger switches at a very affordable price. Once installed on a light post, these triggers will automatically turn on that light whenever they detect another light illuminating nearby. This way, Ronju only needs to manually flip a few switches, and the light from those will activate nearby sensors, gradually illuminating the entire field.
Now, Ronju wonders how many switches he must flip manually to ensure all lights in the field are turned on.

Input

The input begins with an integer **T**, representing the number of test cases.
For each test case:
The first line contains two integers **N** and **M**, where **N** is the number of lights in the field.
The next **M** lines each contain two integers **a** and **b**, indicating that if light **a** is turned on, it will trigger light **b** to turn on as well due to their proximity and other factors. Each test case will be followed by a blank line.

Constraint

$1 \leq N \leq 10^4$
$0 \leq M \leq 10^5$
$1 \leq a, b \leq N$

Output

For each test case, output a single line in the format *Case k: c*, where **k** is the case number starting from 1, and **c** is the minimum number of lights that Ronju must turn on manually before all lights in the entire field are illuminated.

Examples

Input	Output
2 5 4 1 2 1 3 3 4 5 3 4 4 1 2 1 3 4 2 4 3	Case 1: 2 Case 2: 2

G. Zeroing Out Integers

Time: 1 s
Memory: 128 MB

You are given two integers, **a** and **b**. You can perform any number of operations on them (including zero operations). In each operation, you must choose a positive integer **x** and apply one of the following updates:

- Set $a := a - x$ and $b := b - 2x$
- Set $a := a - 2x$ and $b := b - x$

Your goal is to determine if it is possible to make both **a** and **b** equal to 0 simultaneously. You will need to solve **t** independent test cases.

Input

The first line contains an integer **t** ($1 \leq t \leq 100$) — the number of test cases. Each test case consists of one line containing two integers **a** and **b** ($0 \leq a, b \leq 10^9$).

Output

For each test case, output "YES" if it is possible to make both **a** and **b** equal to 0 simultaneously; otherwise, output "NO".

Examples

Input	Output
3 6 9 1 1 1 2	YES NO YES

H. Special Number

Time: 1 s
Memory: 128 MB

We call a positive integer ***special*** if it does not contain two consecutive set bits in its binary representation. Given an integer K, find the K-th smallest special number.

Input

You have to solve the problem for several test cases. Each test contains only one integer, K. The input file terminates with EOF.

Output

Print the K-th special number for each test case.

Examples

Input	Output
3 6	4 9

I. Salary Distribution

Time: 2 s
Memory: 256 MB

You are the leader of a large organization, and you have **n** employees working under you. The number of employees, **n**, is guaranteed to be odd. Your task is to distribute salaries among these employees. You start with **s** dollars available for this purpose, and each employee **i** has a salary range defined by l_i (minimum) and r_i (maximum). The goal is to allocate salaries such that the **median salary is maximized**. To determine the median salary for an odd-length list, you must sort the salaries and select the middle element. For instance:

- The median of the sequence [5, 1, 10, 17, 6] is 6.
- The median of the sequence [1, 2, 1] is 1.

It is guaranteed that you have sufficient funds to pay each employee their minimum salary, meaning that the total of all minimum salaries ($l_1 + l_2 + \ldots + l_n$) will not exceed **s**.

Input

The first line contains an integer $t (1 \leq t \leq 2 \times 10^5)$ — the number of test cases.
For each test case:
The first line contains two integers **n** and **s** ($1 \leq n \leq 2 \times 10^5, 1 \leq s \leq 2 \times 10^{14}$) — the number of employees and the total salary budget. Note that **n** is odd.
The next **n** lines contain two integers each: l_i and r_i ($1 \leq l_i \leq r_i \leq 10^9$), representing the salary range for employee **i**.
It is guaranteed that the sum of all **n** across all test cases does not exceed 200,000.

Output

For each test case, output a single integer — the maximum possible median salary that can be achieved.

Examples

Input	Output
3 3 26 10 12 1 4 10 11 1 1337 1 1000000000 5 26 4 4 2 4 6 8 5 6 2 7	11 1337 6

Notes

- In the first test case, you can allocate salaries as follows: $sal_1 = 12$, $sal_2 = 2$, $sal_3 = 11$. This results in a median salary of 11.
- In the second test case, you must pay the only employee exactly 1337 dollars.

- In the third test case, you can allocate salaries as follows: $sal_1 = 4$, $sal_2 = 3$, $sal_3 = 6$, $sal_4 = 6$, $sal_5 = 7$. The median salary in this case is 6.