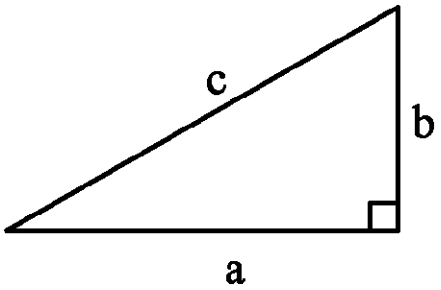
A Right Triangle

Input	Standard Input
Output	Standard Output

A right triangle or right-angled triangle is a triangle in which one angle is a right angle (that is, at 90 degrees angle). The largest side of such a triangle is called hypotenuse. The relation between the sides of the triangle is

$c^2 = a^2 + b^2$, *Where c is the hypotenuse*



Given 3 sides of a triangle, your task is to determine whether the given triangle is a right triangle or not.

Input

The first line has a positive integer `T`, `T <= 100000`, denoting the number of test cases. This is followed by each test case per line.

Each test case consists of a line containing 3 integers `a,b` and `c` denoting the sides of a triangle. All of these sides will be between 1 and 100, inclusive. The sides `a,b` and `c` can be given in any order.

Output

For each test case, the output contains a line in the format `Case #x: M`, where `x` is the case number (starting from `1`) and `M` is "`YES`" when the given triangle is a right triangle or "`NO`" otherwise. Note that the quotes are not required to be outputted.

Sample Input	Sample Output
10 20 16 12 5 3 4 15 12 9 12 5 13 12 13 5 28 82 46 43 96 92 3 4 5 13 5 12 6 10 8	Case #1: YES Case #2: YES Case #3: YES Case #4: YES Case #5: YES Case #6: NO Case #7: NO Case #8: YES Case #9: YES Case #10: YES

B Perfect Cube

Input	Standard Input
Output	Standard Output

A perfect cube is an integer whose cube root is also an integer. For example 1, 8, 27, 64, 125, etc. are examples of perfect cubes but 9, 25 and 113 are not. Given two positive integers A and B, your task is to calculate the number of perfect cubes in the given range between A and B inclusively.

Input

The first line has a positive integer `T`, `T <= 100000`, denoting the number of test cases. This is followed by each test case per line.

Each test case consists of a single line containing two integers `A` and `B` separated by a single space. $1 \leq A \leq B \leq 2{,}000{,}000{,}000$.

Output

For each test case, output contains a line in the format `Case #x: M`, where `x` is the case number (starting from 1) and `M` is the number of perfect cubes in the given range between `A` and `B` inclusive.

Sample Input	Sample Output
10 1 2000000000 42 1011 170 1254 963 2504 282 2430 996 4262 392 2361 293 3308 719 8614	Case #1: 1259 Case #2: 7 Case #3: 5 Case #4: 4 Case #5: 7 Case #6: 7 Case #7: 6 Case #8: 8 Case #9: 12 Case #10: 8

772 5458	

C Mod-3 Permutation

Input	Standard Input
Output	Standard Output

We call a permutation $p_0, p_1, \ldots, p_{n-1}$ of a sequence of integers 0, 1, ... , n-1 mod-3 when for each index i, p_i mod 3 = i mod 3. For example, the permutation 3,1,5,0,4,2 is mod-3 but the permutation 1,2,0,4,5,3 is not. You will be given a permutation. You are required to convert the given permutation into a mod-3 permutationin the minimum number of steps. For each step of the conversion, you need to select two different indices and swap their values.

Input

The first line has a positive integer T, T <= 100000, denoting the number of test cases. This is followed by each test case per line.

Each test case consists of two lines; the first line contains an integer n while the next line contains n integers separated by a single space. These n integers denotes a permutation of 0, 1, .., n-1. n is between 3 and 501 inclusive and is always a multiple of 3.

Output

For each test case, the output contains a line in the format Case #x: M, where x is the case number (starting from 1) and M is the minimum number of swaps required to convert the given permutation into a mod-3 permutation.

Sample Input	Sample Output
10 3 2 0 1 6 1 0 3 2 5 4 6 4 0 3 5 2 4 6 5 3 4 2 0 1 3 0 2 1 9 5 3 8 6 7 0 1 2 4 9 2 1 7 3 5 4 8 6 0 9 3 5 2 0 6 4 7 1 8 9 5 7 1 8 2 6 0 4 3 9 8 3 5 0 6 7 4 2 1	Case #1: 2 Case #2: 3 Case #3: 3 Case #4: 4 Case #5: 1 Case #6: 3 Case #7: 4 Case #8: 2 Case #9: 3 Case #10: 4

D Negative Binary

Input	Standard Input
Output	Standard Output

Number bases have place values that start at 1 (base to the 0 power) and proceed right-to-left to base-1, base-2, and so on. In base-2, the place values are 1, -2, 4, -8, 16, -32, ... (reading from right to left). Thus, counting increasingly from 1 goes like this: 1, 110, 111, 100, 101, 11010, 11011, 11000, 11001, and so on. The advantage of base-2 representation is that you do not have to maintain a sign bit. Your task is to print the given decimal number in base-2.

Input

The first line has a positive integer `T, T <= 100000`, denoting the number of test cases. This is followed by each test case per line.

Each test case contains a single integer decimal number `N` between -10^{17} to 10^{17} inclusive.

Output

For each test case, the output contains a line in the format `Case #x: M`, where `x` is the case number (starting from 1) and `M` is the base-2 representation of the given decimal number.

Sample Input	Sample Output
15 1 -1 2 10 0 100 -100 50 60 1000 -10000 50000 -25000 12345 -23456	Case #1: 1 Case #2: 11 Case #3: 110 Case #4: 11110 Case #5: 0 Case #6: 110100100 Case #7: 11101100 Case #8: 1110110 Case #9: 1001100 Case #10: 10000111000 Case #11: 10100100110000 Case #12: 11100011101010000 Case #13: 1110001110101000 Case #14: 111000001001001 Case #15: 1000100010001000001010100110100000000000000

E Mini Sudoku X

Input	Standard Input
Output	Standard Output

In Mini Sudoku X, there are 6 x 6 boxes to be filled with digits so that each row, column, main diagonal, and 2 x 3 square contains all the digits from 1 to 6. An example of a solution is as follows with the main diagonals shaded:

6	5	1	4	3	2
2	3	4	1	5	6
4	2	5	3	6	1
3	1	6	2	4	5
5	4	2	6	1	3
1	6	3	5	2	4

Write a program that reads a series of 36 digits representing a solution for Mini Sudoku X, and determines whether the solution is correct.

Input

The first line has a positive integer `T, T <= 100000`, denoting the number of test cases. This is followed by each test case per line.

Each test case consists of six lines. Each line of the test case contains six 1-digit integers separated by a space.

Output

For each test case, the output contains a line in the format `Case #x: M`, where `x` is the case number (starting from 1) and `M` is either `0` or `1`. `1` if the test case represents a correct solution and 0 otherwise.

Sample Input	Sample Output
10 1 1 4 3 6 4 3 2 3 5 2 1 4 2 3 2 2 3 2 6 4 3 3 2 2 3 1 2 3 3 4 6 3 1 3 5 2 5 3 3 5 2 2 1 6 4 4 4 5 6 1 1 5 2 1 4 3 4 1 1 6 1 5 3 4 3 4 1 3 2 5 1 2 6 4 3 4 4 1 2 2 4 5 3 1 3 3 2 6 1 2 6 3 2 2 4 4 3 5 1 1 2 4 3 6 2 5 5 4 1 6 2 4 3 3 1 1 1 1 1 1 2 1 1 6 1 4 6 3 5 1 5 5 3 2 2 1 1 4 6 4 1 2 3 2 1 6 1 1 2 3 5 4 2 3 4 3 2 4 2 5 3 3 3 2 4 4 6 4 4 4 3 4 3 4 4 3 1 4 3 3 4 1 6 2 5 6 5 2 4 3 1 1 3 4 2 5 6 5 2 6 1 4 3 2 1 3 5 6 4 4 6 5 3 1 2 3 5 4 2 3 1 5 2 1 3 2 4 4 1 3 6 2 6 2 3 4 1 2 4 3 2 4 4 3 1 5 2 5 4 2 4 3 2 1 3 2 5 2 4 4 4 2 2 1 6 4 5 1 5 6 6 3 3 6 3 1 4 4 6 3 2 3 3 1 2 4 1 1 3 4 4 3 2	Case#1: 0 Case#2: 0 Case#3: 0 Case#4: 0 Case#5: 0 Case#6: 1 Case#7: 0 Case#8: 0 Case#9: 0 Case#10: 0

3 6 1 4 1 1 3 4 5 1 2 5 2 1 5 1 3 2 2 6 3 1 2 2 4 2 2 2 2 1 4 1 3 5 3 4 1 5 4 4 3 3 4 5 1 2 5 1 4 1 5 5 1 2 3 5 3 6 2 6 3 4 6 3 3 5	

F Pair Sum

Input	Standard Input
Output	Standard Output

You are given an integer array of size `N` and an integer `M`. This array has `(N*(N-1))/2` different pairs. You need to calculate how many of those pairs have the sum equal to `M`. For example, if the array is `{1,2,3,4}` and `M` is `5`, then there are exactly 2 pairs `{1,4}` and `{2,3}` whose sum is equal to `M`.

Input

The first line has a positive integer `T, T <= 100,000`, denoting the number of test cases. This is followed by each test case per line.

Each test case consists of two lines. The first line contains 2 integers `N` and `M` separated by a single space. `N` is between `2` and `20,000` inclusive. The second line consists of the `N` integers separated by a single space, denoting the values of the given array. All the numbers in the array are between `1` and `1,000,000,000` inclusive. They are distinct and sorted in increasing order.

Output

For each test case, the output contains a line in the format `Case #x: R`, where `x` is the case number (starting from 1) and `R` is the number of pairs whose sum is exactly the given `M`.

Sample Input	Sample Output
5 8 100 19 25 32 48 52 68 75 81 8 100 19 28 31 49 51 61 72 81 8 100 16 22 38 46 58 62 73 81 8 100 13 21 32 48 52 67 78 87 8 100 13 24 34 43 57 61 76 81	Case #1: 4 Case #2: 3 Case #3: 1 Case #4: 2 Case #5: 2

G	XOR	
	Input	Standard Input
	Output	Standard Output

Given a set of numbers S, consider the following algorithm.
1: Counter = 0
2: Let S' = set of all exclusive-or values between all possible number pairs in S
3: if S=S' goto done
4: Else S = S∪S', counter = counter+1 and goto step 2.
done: print the value of counter.

Given S, your task is to calculate the value of the counter after the program stops.

For example if S = {1,2,4}
Then after step 1: S' = {3,5,6} and S = {1,2,3,4,5,6}, counter = 1
After step 2: S' = {1,2,3,4,5,6,7} and S = {1,2,3,4,5,6,7}, counter = 2
After step 3: S' = {1,2,3,4,5,6,7}
Now no new number is generated. So, we can output the final value of counter, which is 2.

Input

The first line has a positive integer `T, T <= 100,000`, denoting the number of test cases. This is followed by each test case per line.

Each test case consists of two lines. The first line contains `N`, the size of the set. `N` is between `1` and `50` inclusive. The next line contains the `N` integers separated by a single space. These `N` integers are the set S. All of the integers in the set are between `1` and `500,000` inclusive, and distinct.

Output

For each test case, the output contains a line in the format `Case #x: R`, where `x` is the case number (starting from 1) and `R` is the final value of the counter.

Sample Input	Sample Output
10 5 9 16 17 21 20 6 15 18 27 6 21 7 7 9 25 22 10 12 34 33 8 27 22 5 15 25 13 8 31 9 19 4 15 25 21 18 9 22 20 5 33 34 37 36 24 6 4 15 6 14 8 16 7 16 27 41 19 10 26 20 8 16 20 13 11 12 3 24 6 9 24 6 44 35 22 1 26 21 17	Case #1: 2 Case #2: 1 Case #3: 2 Case #4: 2 Case #5: 2 Case #6: 4 Case #7: 3 Case #8: 4 Case #9: 2 Case #10: 3

H Sequence Merging

Input	Standard Input
Output	Standard Output

Initially, we have a given sequence $a_1, a_2, a_3, \ldots, a_n$. In each step, we can take any two adjacent integers a_i and a_{i+1} and replace them with $\max(a_i, a_{i+1})$, thus resulting in a shorter sequence. The cost of this operation is $\max(a_i, a_{i+1})$. After n-1 such steps, the sequence length becomes 1. Given a sequence, your task is to calculate the minimal cost required to make that sequence of length 1.

Input

The first line has a positive integer `T`, $T \leq 100,000$, denoting the number of test cases. This is followed by each test case per line.

Each test case consists of two lines. The first line contains a single integer `N` as the length of the sequence. `N` is between 2 and 1000 inclusive The next line contains `N` integers separated by a single space. Each of the numbers in the sequence is between 1 and 100,000 inclusive.

Output

For each test case, the output contains a line in the format `Case #x: R`, where `x` is the case number (starting from 1) and `R` is minimal cost to reduce the given sequence to a sequence of length 1.

Sample Input	Sample Output
10 6 15 9 8 13 19 17 7 12 15 13 8 12 9 7 8 8 12 14 10 14 18 15 13 9 12 8 3 1 1 1 5 4 7 8 1 6 4 3 2 4 1 6 8 15 16 15 10 18 22 19 22 9 14 9 14 17 21 23 18 17 14 8 13 12 11 5 1 1 6 1 6 8 4 2 1 1 1 8 9 11 12 8 10 11 6 9	Case #1: 75 Case #2: 76 Case #3: 105 Case #4: 42 Case #5: 33 Case #6: 131 Case #7: 147 Case #8: 54 Case #9: 16 Case #10: 76

I Recurrence

Input	Standard Input
Output	Standard Output

Consider a tuple $P_1, P_2, P_3, \ldots, P_n$. Now consider the following recurrence function.

$$F(P_1, P_2, P_3, \ldots, P_n) = \begin{cases} 0 \text{ if any of the } P_i \text{ is negative or the tuple } P \text{ is not sorted in non-increasing order} \\ F(P_1 - 1, P_2, P_3, \ldots, P_n) + F(P_1, P_2 - 1, P_3, \ldots, P_n) + \\ F(P_1, P_2, P_3 - 1, \ldots, P_n) + \cdots + F(P_1, P_2, P_3, \ldots, P_n - 1) \text{ otherwise} \end{cases}$$

F(0, 0, 0…., 0) = 1.

For example, if n is 4 then the value
F(4, 3, 2, -1) is 0 because the last parameter is negative.
F(4, 3, 2, 5) is 0 because the tuple is not sorted from the largest to smallest.
F(4, 3, 2, 1) = F(3, 3, 2, 1)+F(4, 2, 2, 1)+F(4, 3, 1, 1)+F(4, 3, 2, 0)
Given the tuple P, your task is to calculate the value of $F(P_1, P_2, P_3, \ldots, P_n)$. The result can be very big so output the result mod 1,000,000,009 (this is a prime number).

Input

Input starts with an integer *T*, the number of test cases.

Each test case consists of two lines. First line contains n. Second line contains n integers separated by a single space. These are the tuple P. n is between 1 and 1000 inclusive. Each of the numbers in tuple P is between 1 and 1000 inclusive. P will be sorted in non-increasing order.

Output

For each test case, the output contains a line in the format Case #x: R, where x is the case number (starting from 1) and R is the value of

$F(P_1, P_2, P_3, \ldots, P_n) \bmod 1{,}000{,}000{,}009$.

Sample Input	Sample Output
10 3 7 5 4 6 7 7 5 3 2 1 2 4 2 3 7 4 4 4 8 7 5 5 5 7 7 6 5 5 2 8 7 3 6 3 1 4 8 7 4 4 3 6 3 2	Case #1: 100100 Case #2: 398009117 Case #3: 9 Case #4: 25025 Case #5: 923714728 Case #6: 311516464 Case #7: 1430 Case #8: 315 Case #9: 41100051 Case #10: 990

J Division Game

Input	Standard Input
Output	Standard Output

Division game is a 2-player game. In this game, there is a matrix of positive integers with N rows and M columns. Players make their moves in turns. In each step, the current player selects a row. If the row contains all 1s, the player looses. Otherwise, the player can select any number of integers (but at least 1 and each of them should be greater than 1) from that row and then divides each of the selected integers with any divisor other than 1. For example, 6 can be divided by 2, 3 and 6, but cannot be divided by 1, 4 and 5. The player who first makes the matrix all 1s wins. In other words, if in his/her move, player gets the matrix with all 1s, then he/she looses. Given the matrix, your task is to determine whether the first player wins or not. Assume that both of the players will play perfectly to win.

Input

The first line has a positive integer `T, T <= 100,000`, denoting the number of test cases. This is followed by each test case per line.

Each test case starts with a line containing 2 integers `N` and `M` representing the number of rows and columns respectively. Both `N` and `M` are between `1` and `50` inclusive. Each of the next `N` line each contains `M` integers. All these integers are between 2 and 10000 inclusive.

Output

For each test case, the output contains a line in the format `Case #x: M`, where `x` is the case number (starting from `1`) and `M` is "`YES`" when the first player has a winning strategy and "`NO`" otherwise.

Sample Input	Sample Output
5 2 2 2 3 2 3 2 2 4 9 8 5 3 3 2 3 5 3 9 2 8 8 3 3 3 3 4 5 4 5 6 5 6 7 2 3 4 5 6 7 8 9	Case #1: NO Case #2: NO Case #3: NO Case #4: YES Case #5: YES

--- End of the Problem Set ---