ICPC ASIA DHAKA REGIONAL CONTEST 2021

Hosted by

Bangladesh University of Business and Technology

In association with

October 08, 2022

You get 17 Pages, 11 Problems & 300 Minutes

Supported By

Our Sponsors

Media Partner

In Association With

Problem A

Beautiful Blocks Again!

Alice likes playing a game on an **N x M** board where each cell contains some points. Rows are numbered from **1** to **N** from top to bottom, and columns are numbered from **1** to **M** from left to right. She starts from the **(1, 1)** cell and ends her journey on the **(N, M)** cell. But on each move, she can only move in the right or down cell. When she is in a cell, she collects all the points in that cell. Alice loves points a lot. So she always moves in a way that the total number of points she can collect is maximized.

Bob has **an empty N x M** board which is placed **vertically**. He also has **N x M** number of **(1 x 1)** blocks and each block has some points. The blocks fall from **above the board one by one** and he can decide which block falls into which columns. **A block falls as long as it does not hit another block or if it hits the bottom of the board**. **He also does not make a block fall into a full column (a column that already has N blocks)**.

Bob fills the empty board with all blocks in the above-mentioned manner and then gives the filled board to Alice. He wants to make Alice happy as much as possible. **So he fills the empty board with the blocks so that Alice can get the maximum number of points.**

Can you find out how Bob will fill the board and how many points Alice will get?

Input

The first line will contain a single integer **T ($1 \le T \le 10^6$)** denoting the number of test cases. In each test case, the first line will have two space-separated integers **N, M ($1 \le N \times M \le 10^5$)** denoting the number of rows and columns respectively. In the second line, there will be **N x M** space-separated integers **P_1, P_2, ..., $P_{N \times M}$ ($0 \le P_i \le 10^9$)** denoting the points of the blocks. **The blocks fall in the order given in the input.**
The sum of N x M over all test cases does not exceed 10^6.

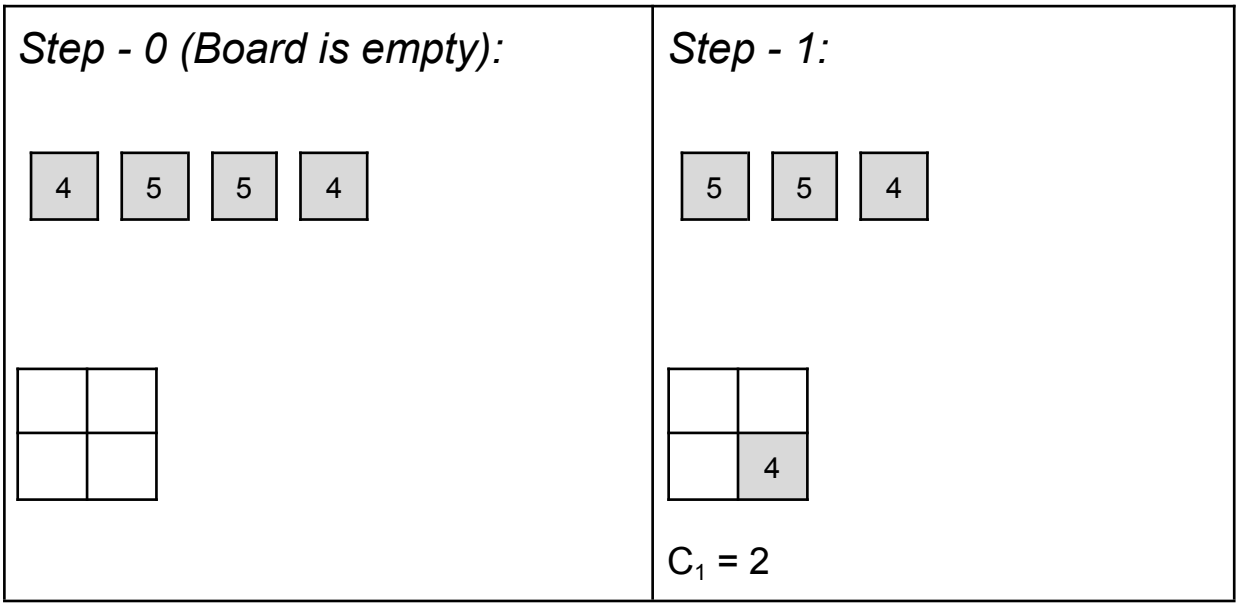
Output

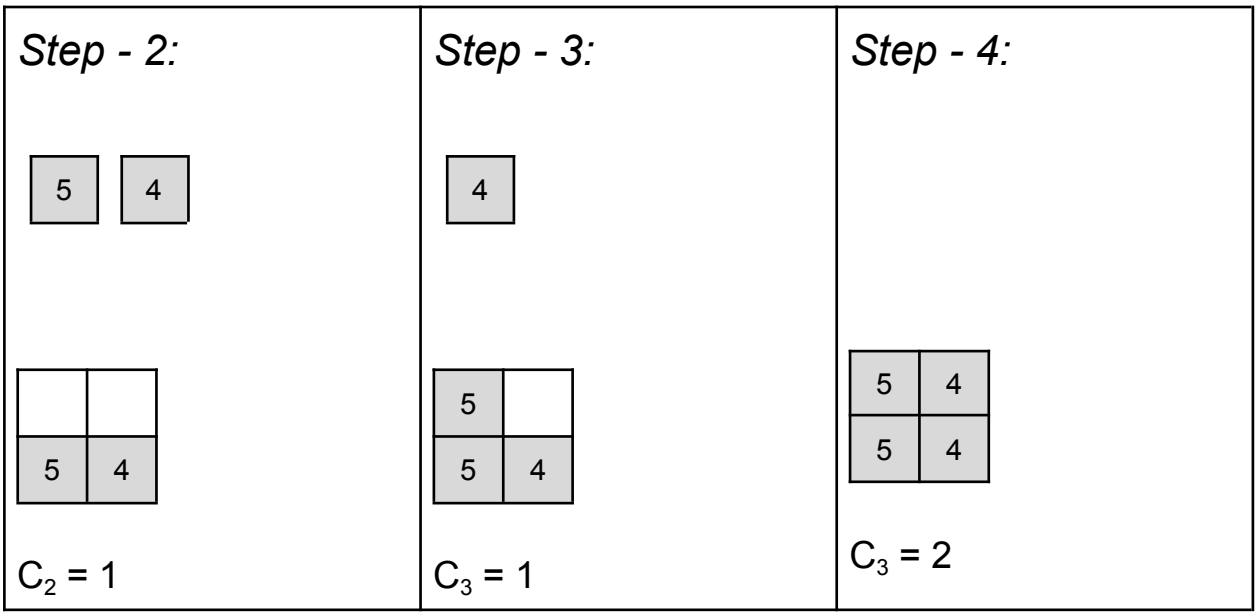
For each test case, in the first line, print the maximum number of points Alice will get. In the second line, print **N x M** space-separated integers, **C_1, C_2, ..., $C_{N \times M}$ ($1 \le C_i \le M$)** where **C_i** denotes the column Bob chooses for the **i^{th}** block to fall. Please see the sample for details.

Note that, there might be multiple optimal ways of falling the blocks. You can output any valid optimal solution.

Sample Input	Output for Sample Input
2 2 2 4 5 5 4 2 1 10 9	14 2 1 1 2 19 1 1

Explanation: For the 1st sample test, initially 2 ×2 board is placed vertically. Blocks will fall into it one by one maintaining the given input order. An example of filling the board with blocks so that Alice can maximize her points is shown below.





One possible optimal path visited by Alice is shown below (visited cells in bold).

5	4
5	**4**

So the maximum number of points obtained by Alice = 5 + 5 + 4 = 14.

Problem B

Black Bishop

We have an **n** × **n** sized checkerboard where every cell of the board is colored either black or white so that no two cells of the same color are adjacent. The board size, **n**, will always be **odd** and all the corner cells of the board are black.

We have **m** identical bishops at the black cells on the board, where **m** is at most **n**. We are given the initial and target configuration of the bishops. You have to move the bishops from the initial configuration to the target configuration in at most **4**×**m** moves.

In a move a bishop can only move diagonally but not over another bishop. Two bishops can never be in the same cell at the same time.

Input

The input begins with the number of cases **t (1 ≤ t ≤ 100)**. Then **t** test cases follow. Every case starts with **n (1 ≤ n ≤ 99,999)** and **m (1 ≤ m ≤ n) and n is odd**. Then follows **m** lines describing the initial bishop configuration. Next **m** lines describe the target configuration. Each line of the configuration contains two integers **r, c (1 ≤ r, c ≤ n)** where they denote the row and column number of the cell a bishop is at.

Sum of **n** across all cases will not exceed **100,000**.
No two bishops in the initial configurations will be in the same cell. Same goes for the target configuration.

Output

For each case, output the case number and the number of moves denoted by **k**. Next **k** lines contain **4** integers **r1, c1, r2, c2 (1 ≤ r1, c1, r2, c2 ≤ n)** meaning a bishop is moved from **r1, c1** position to **r2, c2** position (They can be same cell as well). Check the Sample input/output for further clarity. Also please be careful with the output format. Unintended whitespaces might cause a wrong answer.

Sample Input	Output for Sample Input
2 5 2 1 3 5 3 3 1 3 5 1 1 1 1 1 1	Case 1: 2 1 3 3 5 5 3 3 1 Case 2: 0

Problem C

Codovid Virus

The world is stunned by the CodoVid virus, which is believed to be one of the deadliest viruses the world has ever seen. Due to its self-reproduction mechanism, the DNA sequence of the CodoVid virus grows at a high speed. The DNA sequence of CodoVid virus is a binary string consisting of **0's** and **1's** only. A codovid virus has an initial DNA sequence S_1. Starting from day 2, each day codovid virus reproduces a new sequence and adds the new sequence to its DNA sequence.

The sequence of DNA added on the i[th] day, $S_i = Reverse(S_{i-1}) + Flip(S_{i-1}) + Reverse(S_{i-1})$,

- The "+" symbol represents $string\ concatenation$ operation.
- $Reverse\ ()$ function returns the characters in reverse order. i.e. $Reverse\ (1100) = 0011$
- $Flip()$ function changes all the 0's to 1's and vice versa. i.e. $Flip\ (1001) = 0110$

So, on the first day the Codovid DNA sequence is S_1. On the second day the sequence is S_1+S_2. On the third day, the sequence will become S_1+S_2+S_3. So, it is an ever-growing sequence, "S_1+S_2+S_3+……."

An example will make it clear.

Let's assume the Initial CodoVid DNA sequence on Day 1, $S_1 = 100$.

So, $S_2 = Reverse(100) + Flip(100) + Reverse(100) = (001) + (011) + (001) = 001011001$,

$S_3 = Reverse(001011001) + Flip(001011001) + Reverse(001011001) = 100110100110100110100110100$

So, the first few characters of the CodoVid DNA sequence is,

$$100\ 001011001\ 100110100110100110100110100\ldots\ldots\ldots$$

So, the 1[st] character of the codovid DNA sequence is '1'. Similarly, 2[nd], 3[rd], 4[th], 5[th] is '0', 6[th] character is '1' and 7[th] character is '0' and so on.

In this problem, you are given the initial DNA sequence of the Codovid virus, S_1 and two integers L and R. You have to count the number of 1's from the L[th] character to the R[th] character of the Codovid DNA sequence.

Input

The input begins with an integer **T (1 ≤ T ≤ 100)** denoting the number of test cases. T test cases follow. Every case starts with a non-empty string $\mathbf{S_1}$ (**1 ≤ |S_1| ≤ 10^4**) in a single line. Next line of the input is an integer **Q** (**1 ≤ Q ≤ 5000**) denoting the number of queries. Then follow Q lines containing two integers **L, R** (**1 ≤ L ≤ R ≤ 10^{17}**). You may safely assume that, **Sum of length(S_1) over all test cases ≤ 5 x 10^5.**

Output

For each test case, Print the case number in the format "**Case X:**" in a single line (without quotes), where **X** is the case number. Then, for each query of that test case, print the desired answer in a single line.
Please check the samples for further clarification.

Sample Input	Output for Sample Input
1 100 4 1 3 4 12 10 20 1 100	Case 1: 1 4 5 48

Problem D

Der Stern

Let's imagine our earth is at the origin of a 3d cartesian coordinate system and has a radius of 100 units. The most beautiful star of the universe *Der Stern* is at (0, 0, s). You can imagine Der Stern to be a tiny but very bright point. Tonight the moon is at (0, 0, m) and the radius of the moon is r. Calculate the probability that a person would be able to see Der Stern if the position of the person is randomly chosen on the earth surface. You will be able to see Der Stern if there is a line of sight between the person and Der Stern.

Please see the sample Input output for more clarity.

Input

The input begins with the number of cases T ($1 \le T \le 300$). T test cases follow. Every case consists of 3 integers: s, m, r ($-1{,}000{,}000 \le s, m \le 1{,}000{,}000$; $1 \le r \le 1{,}000{,}000$).

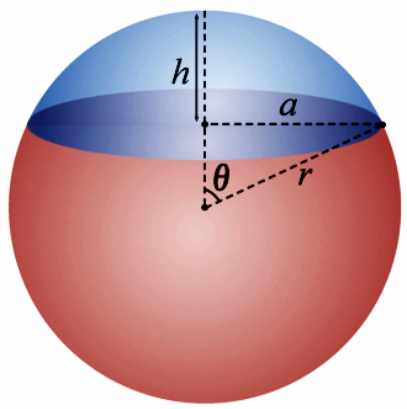
The earth, moon and the Star would always be disjoint. That is, they won't intersect or touch, also none of them will be inside the other.

Output

For each case, output the case number and the required probability. Answer is expected to be within absolute or relative error of 0.001 of the correct solution.

Sample Input	Output for Sample Input
1 1000 500 1	Case 1: 0.4499189902

Note



In the picture, the sphere has a radius of r. Suppose we cut a cap of height h ($h \le r$). The surface area of the cap is $2\pi rh$.

Problem E

Calendars

You will be given a description of several different calendars . You will have to convert the date of one calendar to the other one. If there are p months in a calendar then the calendar is described by a positive integer p followed by p positive integers. These p integers denote the number of days in each month. The months are identified by integers 1, 2, …p. You do not need to worry about leap years etc as there are no such things associated with these calendars. **Please note that for this problem days, months and years are 1-indexed and these calendars start from the same day (The very first day of all calendars 1-1-1 is the same day).**

Input

First line of the input file contains a positive integer **NC** not exceeding **210**. This denotes the number of calendar descriptions to follow. Each of the next **NC** lines describe a Calendar. These calendars are identified as Calendar **1**, **2** …, **NC** respectively.

The description of each calendar is given in a single line. The line starts with an integer **p (0 < p ≤ 20)** which denotes the number of months in the calendar followed by p positive integers m_1, m_2, …, m_p, here m_i denotes the number of days the i-th month has. None of these **p** integers exceed the value **50**.

The next line contains an integer **q (0 < q ≤ 1010)** which denotes the no of queries. Each of the next **q** lines contain a single query.

Each query has five integers: **cal_1 (1 ≤ cal_1 ≤ NC)**, **cal_2 (1 ≤ cal_2 ≤ NC)**, **day**, **month** and **year (1 ≤ year ≤ 5000)**. Here day, month, year denotes a date in cal_1 and the task is to convert it to the corresponding date in **cal_2**. You can assume that no invalid date will be given as input.

Output

For each query your program should produce one line of output. This out contains three integers **d_2, m_2, y_2** which denotes the day, month and year of the date when converted to **cal_2**.

Sample Input	Output for Sample Input
2 3 10 20 30 4 20 30 10 20 2 1 2 18 2 2 2 1 17 4 1	8 1 2 7 2 2

Problem F

Flip

Bitstring is a string consisting of only 0 and 1. Each element in the string is called a bit. A bitstring is called circular when the first and last bit are considered to be adjacent.

Given a circular bitstring of size **n** and a number **k**. You want to make all the bits zero. There are two possible operations.

(i) Take a bit and flip it, the cost of this operation is **1**.
(ii) Take any **k** consecutive bits and flip them all, the cost of this operation is **0**.

You are also given **q** updates. In each update one bit will be flipped. After each update you have to print one line, the minimum cost to make all the bits zero in the current bitstring.

Input

Input starts with **3** numbers, **n**, **k,** and **q** respectively. The following line contains a bitstring of length **n**. Then **q** lines each containing an index between **1** to **n**, denoting which bit to flip.

Constraints

- $2 \le k \le n \le 3 \times 10^5$
- $1 \le q \le 3 \times 10^5$

Output

Print one line for each update containing the minimum cost. Please see the sample for details

Sample Input	Output for Sample Input
5 3 2 10010 5 1	0 0

Explanation

Initially the bitstring is: 10010
After the first update it becomes, 10011
We can flip 4'th, 5'th and 1st bit simultaneously with the 2nd type of operations and make every bit 0 with cost 0.
After the 2nd update it becomes, 00011
Now we do the 2nd type of operations 4 times to achieve all 0 states.

0**001**1 → 01101
011**01** → 11110
1111**0** → 00111
00**111** → 00000

Note

Flipping a bit means,

i) If the bit is 0, change it to 1.
ii) If the bit is 1, change it to 0.

Problem G

Make GCD Great Again!

You are the president of the United Communities of Contest Programming. In order to maintain the reputation of UCCP, you need to solve a simple, yet interesting problem. You are given an array **A** of **N** positive integers. You can perform the following operation on the given array some number of times (maybe no operation at all), **but not more than once** on a particular element of the array:

- Choose an index **i** of the array. Change $\mathbf{A_i}$ to some positive number $\mathbf{X > A_i}$. The cost of this operation is $\mathbf{|A_i + X|} \times \mathbf{|A_i - X|}$.

You have to perform the operation to make the GCD (Greatest Common Divisor) of all the elements of the array greater than **1**. But you have to do it at the minimum total cost. You also need to find the maximum GCD you can make with the minimum total cost. Can you make the GCD great again?

Input

The first line will contain a single integer **T**. Each test case will have a single integer **N**, denoting the number of elements of array **A**. The following line will contain **N** space separated integers $\mathbf{A_1}$, $\mathbf{A_2}$, ..., $\mathbf{A_N}$ denoting the elements of array **A**.

Output

For each case, print one line with "**Case <x>: <y> <z>**", where **x** is the case number, **y** is the minimum total cost to make the GCD of all the elements of array **A** greater than **1** and **z** is the maximum GCD you can make with the minimum total cost.

Constraints

- $\mathbf{1 \le T \le 20}$
- $\mathbf{1 \le N \le 10^5}$
- $\mathbf{1 \le A_i \le 10^6}$

Sample Input	Output for Sample Input
2 5 3 6 12 15 21 7 5 13 17 20 25 33 30	Case 1: 0 3 Case 2: 191 2

Problem H

Vishon Xor

You are given an array **A** of length **N**. You can perform two kinds of operations on this array any number of times (possibly zero).

1. Choose any non-negative integer **X** and then for each **i**, replace **A[i]** with **(A[i]** $\oplus$ **X).**
2. Sort the array **A**.

Cost of the array is $min_{i=1}^{n-1} abs(A[i] - A[i+1])$. You need to minimize the cost.

Input

First line will contain a single integer **T** representing the test cases. For each test case there will be a single integer which will represent **N**. In the next line there will be **N** space separated integers where the i-th integer will represent **A[i]**.

Constraints

- **1 ≤ T ≤ 2 x 10^5**
- **1 ≤ N ≤ 2 x 10^5**
- **sum of N over all test cases ≤ 2 x 10^5**
- **0 ≤ A[i] ≤ 10^{18}**

Output

For each test case print a single line representing the minimized cost with test case number (Please check sample output for output format).

Sample Input	Output for Sample Input
2 3 1 2 3 3 2 2 1	Case 1: 1 Case 2: 0

Problem I

Keep Calm, Keep padding

People of Wakanda really love biking. Each city has their own bike hub. You can pick a bike from any hub! After arrival at your destination city, you just need to return the bike to the bike hub.

On a typical morning, each bike hub has **K** bikes. Whole day people move randomly. So at the end of the day the number of bikes in the hubs gets changed. It can be **greater than, less** than or equal to **K**. We need to re-distribute the bikes so that it is the same as in the morning. So after shuffling each bike hub should have exactly **K** bikes again.

Wakanda country consists of **N** cities and there are **N-1** bi-directional roads between the cities. Every city is reachable from every city by roads.

The Strawhat company is responsible for this bike management system. They have a giant truck of unlimited capacity to move bikes. Their policy is that they will visit all roads exactly twice, but can visit any city multiple times.

Now you need to find **any possible route** so that all conditions are met. You can start from any city. From any bike hub you can collect any numbers of the bikes and similarly give any numbers of bikes from the truck!

Input

The First line of the input contains a single integer **T (1 ≤ T ≤ 100)** denoting the number of test cases. The description of **T** test cases as follows:

- The first line of each test case contains two integers **N (1 ≤ N ≤ 10000)** and **K(1 ≤ K ≤ 1000)**.
- Each of the next **N−1** lines contains two space-separated integers **x** and **y** denote that cities **x** and **y** are connected by a road.
- The last line contains **N** integers denoting the city **i** has $\mathbf{x_i}$**(0 ≤** $\mathbf{x_i}$ **≤ 10000000)** bikes. It is guaranteed that sum of all $\mathbf{x_i}$ will be **N**×**K**.

Output

For every test case, you must print the case number, followed by "**YES**" if there is a possible route, otherwise "**NO**".
If there is a possible route, then print **2N-1** additional lines describing the order of the road traversal. Each line should have 2 integers: **u** and **p.**
Where, **u** indicates the city you will go to and **p** indicates how many bikes you collect from or give to the bike hub of this city . Non-negative **p** means you are putting **p** bikes to the bike hub and negative **p** means you are taking bikes from bike hub. First line indicates you start from this city.

Please check the samples for further clarification.

Sample Input	Output for Sample Input
2 2 100 1 2 200 0 5 20 1 4 2 4 4 5 3 4 31 7 22 23 17	Case 1: YES 2 0 1 -100 2 100 Case 2: YES 4 -13 2 13 4 -3 5 3 4 0 1 -11 4 11 3 -2 4 2

Explanation

In the first case, we start from the bike hub of city 2 and do nothing. Then go to the bike hub of the city 1 and take 100 bikes from there. After this, return to city 2 and put 100 bikes on the bike hub.

Similarly in the second case, we start from the bike hub of the city 4 and take 13 bikes. Then move to city 2 and give 13 bikes and so on.

Problem J

ICPC Standing

Students get the chance to interact with different institutions while showcasing their problem-solving, programming, and teamwork abilities through the International Collegiate Programming Contest (ICPC). The ranking of teams for this contest will be decided based on the following rules.

- The team with the higher number of solved problems will get a higher rank. Rank 1 is considered the highest rank.
- In case of a tie between two teams with the same number of problems solved, the higher rank is determined by penalty time. The team with a lower penalty time will get the higher rank.

The web page that shows all team's ranks in the contest from higher rank to lower rank, is known as the contest standings page. Usually, the contest standing page shows a live rank list consisting of the team name, number of problems solved, total penalty time, problem status (solved or unsolved), number of attempts on each problem, and so on. An example rank list page is shown in the following figure.

STANDINGS

Rank	Team Name	Solve/ Penalty	A	B	C	D	E	F	G	H	I	J	K
1	Team X (ABC University)	10 1394	1 (7)	2 (62)	1	3 (130)	1 (34)	3 (299)	1 (63)	2 (174)	2 (235)	1 (140)	1 (110)
2	Team Y (University of DEF)	9 688	1 (3)	1 (28)	9	1 (18)	1 (54)	4	1 (105)	2 (174)	1 (147)	1 (67)	2 (52)
3	Team Z (ABC University)	9 1552	1 (4)	1 (92)		1 (9)	2 (71)	1	4 (166)	4 (247)	3 (226)	1 (169)	5 (308)
4	Team A (XYZ University)	8 826	1 (4)	1 (101)		1 (72)	1 (52)	1	1 (156)	2 (212)	3	1 (134)	1 (75)

This year the organizer will not provide any rank list in the contest standing page.

The contest is running. There are still 3 hours remaining in this contest. As a contestant, you are only informed that there are **P** problems in the problem set, your team has already solved **S** problems, and your current team rank is **R**. You have to determine whether there is a chance by any means for your team to become champion after the end of the contest.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 18150)**. Each test case will contain three integers **P (1 ≤ P ≤ 10)**, **S (0 ≤ S ≤ P)** and **R (1 ≤ R ≤ 165).**

Output

For each test case, Print the case number in the format "**Case X:**" in a single line (without quotes), where **X** is the case number, followed by "**Yes**" if there is a chance for your team, "**No**" otherwise.

Sample Input	Output for Sample Input
2 10 0 165 10 10 1	Case 1: Yes Case 2: Yes

Problem K

Clash of Coprimes

Roh has recently started working on a new project and he really enjoys working on it. The thing that annoys him, however, is that the requirements seem to keep changing continuously. Everyday, Roh's supervisor Zach comes to him with a new list of requirements that Roh must fulfill. The ever-changing requirements have frustrated Roh and he wishes for once that Zach could make up his mind.

Roh's task is simple. He must construct an array $a[1 \ldots n]$ of size n, whose elements are positive integers **greater than 1**. Everyday Zach comes to Roh with a new set of requirements. The requirement on the **i**th day is defined by a subset of indices, S_i. For any two indices **x, y** such that **x** $\in S_i$ and **y** $\notin S_i$, it is required that $gcd(a[x], a[y]) = 1$.

On the **i**th day, Roh must construct an array of positive integers $a[1 \ldots n]$ of size **n**, such that,

- $a[i] \geq 2$ for all $1 \leq i \leq n$.
- The array satisfies all requirements from day 1 to day **i**, ie. it satisfies each of the sets $S_1, S_2, S_3, \ldots, S_i$

Of course, there might be many such arrays and Roh must find the array with the minimum sum of elements. Zach doesn't have the time to go through all of Roh's work. So, he asks Roh to report only two numbers, the minimum sum of elements that Roh can achieve, and the number of valid arrays with the minimum sum.

Help Roh complete his task.

Input

The first line contains **T**, the number of test cases. The first line of each case contains two integers **n**, and **q** denoting the number of elements and the number of requirements. It is followed by **q** lines. The **i**th line describes the set S_i. It starts with an integer **k** denoting the number of elements of S_i. Then **k** distinct integers follow $S_{i,1}, S_{i,2}, S_{i,3}, \ldots, S_{i,k}$ the elements of S_i.

Constraints

- $1 \leq T \leq 100$
- $2 \leq n \leq 10^5$
- $1 \leq q \leq 10^5$
- $1 \leq k \leq n$
- $1 \leq S_{i,j} \leq n$ for all $1 \leq j \leq k$
- All elements of a set S_i are distinct, ie. $S_{i,1}, S_{i,2}, \ldots, S_{i,k}$ are distinct.
- Sum of **n** over all cases does not exceed 5×10^5.
- Sum of **q** over all cases does not exceed 5×10^5
- The total number of elements over all sets over all test cases does not exceed 10^6.

Output

For each test case, print **q** lines. The **i**th line for each case should contain two space-separated integers. The first integer is the minimum sum of a valid array that fulfills all requirements upto day **i**. The second integer is the number of valid arrays with the minimum sum. Since the number of valid arrays can be large, print the number of valid arrays modulo **998244353**.

Sample Input	Output for Sample Input
1 4 2 2 1 2 3 1 2 3	10 2 12 2

For the first day, **2, 2, 3, 3** and **3, 3, 2, 2** are the interesting sets with the minimum sum.
For the second day, **2, 2, 3, 5** and **2, 2, 5, 3** are the interesting sets with the minimum sum.

A Problem A

Omicron Juice

I have three kids- Alpha, Beta and Gamma. Every morning I need to prepare them for school, serve them breakfast and say them bye. You might think that I don't like all these. But to be honest I do like it, specially I have fun serving them breakfast. Every morning I serve them Omicron Juice. But I can't always pour the juice in equal amounts. If they don't get equal amounts of juice then they start fighting. So I try my best to make them equal. To be more specific…

There are three glasses with juice of amount **A**, **B** and **C** units. I have another small empty glass which can hold **1** unit of juice. Using this small glass I can take **1** unit of juice from one glass and pour it to another glass. Can you figure out if they will go to the school peacefully or are they going to start a fight? I promise if there is any possibility for a peaceful morning I will definitely make that happen! Just to clarify, the morning would be peaceful if all those three glasses contain equal amounts of juice, the extra **1** unit glass is empty and during the entire process you do not spill any juice. Also assume that the kids glass can hold any amount of juice.

Input

The input begins with the number of cases **T (1 ≤ T ≤ 10,000)**. In each of the following **T** lines you will find three non negative integers: **A, B, C** (each of at most **20)**.

Output

For each case, output the case number and print if the morning is **"Peaceful"** or if they are going to have **"Fight"**. Please see the sample for details.

Sample Input	Output for Sample Input
2 3 5 1 3 4 1	Case 1: Peaceful Case 2: Fight

B Problem B

Omicron Juice Again

I have three kids- Alpha, Beta and Gamma. Every morning I need to prepare them for school, serve them breakfast and say them bye. You might think that I don't like all these. But to be honest I do like it, specially I have fun serving them breakfast. Every morning I serve them Omicron Juice. But I can't always pour the juice in equal amounts. If they don't get equal amounts of juice then they start fighting. So I try my best to make them equal. To be more specific…

There are three glasses with juice of amount **A**, **B** and **C** units. I have another empty glass which can hold exactly **K** units of juice (after all I am Kappa!). In a move using the extra glass, I can transfer **K** units of juice from one of the kids glasses to another kids glass as long as the source glass has at least **K** units of juice. If I tell you **A**, **B**, **C** and **K**, can you figure out if they will go to the school peacefully or are they going to start a fight? I promise if there is any possibility for a peaceful morning I will definitely make that happen! Just to clarify, the morning would be peaceful if all those three glasses contain equal amounts of juice, the extra **K** unit glass is empty and during the entire process you do not spill any juice. Also assume that the kids glass can hold any amount of juice.

Input

The input begins with the number of cases **T (1 ≤ T ≤ 200,000)**. In each of the following **T** lines you will find four non negative integers: **A, B, C, K (each of at most 20)**. **K** is a positive integer.

Output

For each case, output the case number and print if the morning is **"Peaceful"** or if they are going to have **"Fight"**. Please see the sample for details.

Sample Input	Output for Sample Input
2 3 5 2 2 3 9 6 3	Case 1: Fight Case 2: Peaceful

Note: You might have to use faster I/O.

C Problem C
Bitonic Beaver

A permutation, $P[1 \cdots n]$ of length n is a sequence of integers where each integer from 1 to n appears exactly once.

A permutation is called Bitonic if it is first increasing then decreasing. Formally, A permutation $P[1 \cdots n]$ is bitonic if there exists $1 \leq i \leq n$ such that, $P[1] < P[2] < \cdots P[i]$ and $P[i] > P[i+1] > \cdots P[n]$.

For example, [1, 2, 4, 3], [3, 4, 2, 1] and [1, 2, 3, 4] are bitonic sequences. In particular, increasing or decreasing sequences are bitonic as well.

A permutation P is said to be compatible with another permutation Q if $P[Q[1]], P[Q[2]], P[Q[3]], \cdots P[Q[n]]$ is bitonic.

For example, Let P = [2, 1, 4, 3] and Q = [1, 4, 3, 2]. Then,

$P[Q[1]], P[Q[2]], P[Q[3]], \cdots P[Q[n]]$
$= P[1], P[4], P[3], P[2]$
$= 2, 3, 4, 1$

which is bitonic, so, P is compatible with Q.

Mrs Beaver has k permutations, $Q_1, Q_2, Q_3, \cdots, Q_k$ each of length n. She wants to find the number of permutations P (of length n) such that P is compatible with all $Q_i (1 \leq i \leq k)$. Since this number can be large, she is happy to find its value modulo 998244353.

Input

The first line will contain a single integer $T (1 \leq T \leq 10^5)$.

Each test-case starts with a single line containing n and k $(1 \leq n, k \leq 10^6)$. Each of the next k lines contain n integers each representing a permutation Q_i. Each integer from 1 to n appears exactly once in each line.

The sum of nk over all test cases does not exceed 10^6.

Output

For each case, output a single integer, the number of valid permutations modulo 998244353.

Sample Input	Output for Sample Input
2 4 2 2 1 4 3 3 4 1 2 5 3 2 1 4 5 3 4 1 5 3 2 3 1 4 2 5	8 0

D Problem D

Lazy Squirrel

There is a rooted tree of **N** nodes numbered from **1** to **N** and **1** is the root of the tree. Each node contains a card on which there is an English letter written in Capital. There is a squirrel who wants to stay in this tree tonight.

The squirrel believes staying in a node **X** (**1 ≤ X ≤ N**) makes him happy, if the letters written in the cards on nodes in the shortest path from root to **X** forms a palindromic sequence. Now the squirrel wants to know what is the probability of being happy, if he randomly chooses a node for staying tonight. Since the squirrel is lazy, he wants your help.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 100)**. Each test case starts with an integer **N (1 ≤ N ≤ 10^6)**, denoting the number of nodes of the Tree. The following line will contain a string **S**, where i'th character of **S** represents the letter written on the card of i'th node (**|S| = N ,'A' ≤ S_i ≤ 'Z'**). Next **N - 1** lines will contain two integers **U** and **V (1 ≤ U, V ≤ N, U ≠ V)** indicating there is an edge between **U** and **V**. Summation of **N** over all test cases **≤ 10^6**.

Output

For each case, print the case number and the result in **P/Q** format. Where **P** is the numerator and **Q** is the denominator. **P** and **Q** should be relatively prime. See the samples for details.

Sample Input	Output for Sample Input
2 5 ABAAC 1 2 1 3 2 4 2 5 1 A	Case 1: 3/5 Case 2: 1/1

Explanation:

For the first case, there are three possible nodes where the squirrel will be happy to stay tonight.

1. X = 1 (A , 1)
2. X = 3 (AA , 1->3)
3. X = 4 (ABA , 1->2->4)

E Problem E

Pairdrome

A palindrome is a string if it reads the same backward as forward, for example, `"a"`, `"aba"`, and `"axddxa"` are palindromes but `"ab"`, `"icpc"`, and `"dhaka"` are not. And a substring of a string is a contiguous subsequence of that string, for example, `"i"`, `"c"`, `"cp"`, and `"icpc"` are some of the substrings of `"icpc"`.

In this problem, you are given two strings X and Y consisting of lowercase English letters. You need to calculate the number of ways you can form a pairdrome from those strings. A pairdrome is a palindrome and it is a concatenation of two substrings, one from X and another from Y.

More formally, suppose you have two strings X and Y. A string T is a pairdrome if

1. T is a palindrome.
2. $T = X_{[a, b]} + Y_{[c, d]}$, where $1 \leq a \leq b \leq |X|$ and $1 \leq c \leq d \leq |Y|$ and $S_{[i, j]}$ is a substring of S starting from the index i and ending at index j and $1 \leq i \leq j \leq |S|$. (We are using 1-based indexing, here $|S|$ denotes the length of the string S and the plus $(+)$ sign denotes string concatenation.)

For example, suppose X is `"kan"` and Y is `"zan"`. Here we have 5 pairdromes, they are

1. $X_{[2, 2]} + Y_{[1, 2]} = "a" + "za" = "aza"$
2. $X_{[2, 2]} + Y_{[2, 2]} = "a" + "a" = "aa"$
3. $X_{[2, 3]} + Y_{[2, 2]} = "an" + "a" = "ana"$
4. $X_{[3, 3]} + Y_{[2, 3]} = "n" + "an" = "nan"$
5. $X_{[3, 3]} + Y_{[3, 3]} = "n" + "n" = "nn"$

Since the answer can be quite large, you need to print the answer modulo $10^9 + 7$.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 100)**. Each test case will contain two strings **X** and **Y** separated by a space in one line. These strings consist of lowercase English letters. The length of each string will be less than or equal to 10^5. And in a test file, the summation of all the string lengths will not exceed 2×10^5.

Output

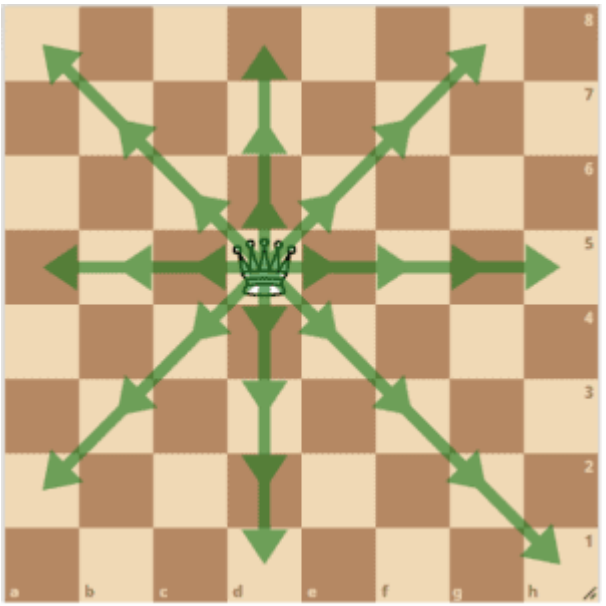
Print the case number followed by the answer to that test case in this format "**Case X: Y**" (without quotes), where **X** is the case number and **Y** is the number of ways for forming a pairdrome modulo **$10^9 + 7$.** Please see the sample for details.

Sample Input	Output for Sample Input
4 kan zan dhaka taka meow max x y	Case 1: 5 Case 2: 20 Case 3: 2 Case 4: 0

F Problem F

A Lonely Queen

We have a chessboard with **n** rows and **m** columns. The queen is standing at the **S_y-th column** of the **S_x-th row,** i.e. **(S_x, S_y)**. She wants to move to the cell **(D_x, D_y).** The queen can move in 8 directions at any time. The cost of moving in the i-th direction is **C_i.** Formally, from the cell (A_x, A_y), she can perform the following moves:



1. Move to the cell (A_x - k, A_y - k) with C_1 cost.
2. Move to the cell (A_x - k, A_y) with C_2 cost.
3. Move to the cell (A_x - k, A_y + k) with C_3 cost.
4. Move to the cell (A_x, A_y - k) with C_4 cost.
5. Move to the cell (A_x, A_y + k) with C_5 cost.
6. Move to the cell (A_x + k, A_y - k) with C_6 cost.
7. Move to the cell (A_x + k, A_y) with C_7 cost.
8. Move to the cell (A_x + k, A_y + k) with C_8 cost.

Here, **k** can be any positive integer provided that the queen does not go out of the chessboard.

There are **q** cells, each of which has an obstacle. The queen cannot go to a cell containing an obstacle or go over an obstacle. There are also **p** one-way magic tunnels. A tunnel can be defined by two cells (**U_x, U_y)**, (**V_x, V_y)** and an integer **W**. This means that if the queen is at cell **U** currently, she can move to cell **V** using this tunnel with cost **W**. A cell can have any number of tunnels originating from or ending at it. Regardless of the positions of the endpoints of the tunnels and the obstacles, the queen can always use a tunnel to move from the starting cell **U** of the tunnel to the cell **V** at the end of the tunnel. The endpoints of the tunnels, cell **S** and cell **D** do not have any obstacles.

You need to find out the minimum cost for the queen to move from cell **S** to cell **D**.

Input

The first line will contain a single integer **T** denoting the number of test cases**.** Each test case will have four integers **n, m, p,** and **q** in the first line. The next line will contain four integers **S_x, S_y, D_x,** and **D_y,** denoting the source cell **S** and the destination cell **D.** The next line will contain **8** integers where the i-th integer denotes the value of **C_i**. Each of the next **p** lines will contain 5 integers **U_x**, **U_y, V_x, V_y,** and **W,** denoting the magic tunnels. The following **q** lines will contain 2 integers **B_x** and **B_y** denoting a cell with an obstacle.

Constraints

- **1 ≤ T ≤ 100**
- **1 ≤ n,m ≤ $5x10^4$**
- **0 ≤ q ≤ (n x m)**
- **0 ≤ Σp over all test cases ≤ $5x10^4$**
- **1 ≤ row of any cell ≤ n**
- **1 ≤ column of any cell ≤ m**
- **1 ≤ W ≤ 10^9**
- **1 ≤ C_i ≤ 10^9**
- **1 ≤ $\Sigma(n \times m)$ over all test cases ≤ 10^5**

Output

For each case, print the case number and the answer in a single line. If it is not possible for the queen to reach the destination cell from the starting cell, the answer should be "**-1**". Please see the sample for details.

Sample Input	Output for Sample Input
3 1 4 4 0 1 1 1 2 4 6 8 1 5 1 9 1 1 1 1 3 1 1 2 1 1 5 1 2 1 2 4 1 2 1 1 6 4 5 0 5 4 5 3 1 40 1 97 70 47 1 25 80 2 1 3 3 3 4 3 5 4 4 4 5 2 5 4 5 3 1 40 1 97 70 47 1 25 80 4 5 3 2 1 2 3 2 5 27 2 1 3 3 3 4 3 5 4 4	Case 1: 2 Case 2: -1 Case 3: 3

G Problem G

Alice, Bob and Tree

Alice and Bob met together after a long time. As a token of appreciation, Bob presented Alice a colorful gift box. After opening the box, she was amused by a beautiful tree with the following specification.

- A tree has N vertices and N-1 edges. All the vertices of the tree are connected.
- Each of the vertices has an associated weight with it, the weight of the i^{th} vertex is W_i.
- The distance between any two adjacent vertices is 1

Now, Bob asked Alice to choose the tree's root vertex in such a way that the value of the following formula is maximized.

$$\sum_{i=1}^{N} \left((W_{root} - W_i) \times (-1)^{Distance(root, i)} \right)$$

If there are several such vertices for which the value of the above formula is maximized, then Alice has to choose the root with the smallest label. If Alice can answer the question successfully, Bob promised to gift her a colorful balloon. Since Alice is not good at solving problems, she asked for your help. Can you solve this problem for her?

Input

Input starts with an integer **T** denoting the number of test cases. Each of the **T** test cases will start with an integer **N** denoting the number of vertices in a single line. Next line will contain **N** space-separated integers $W_1, W_2, \ldots, W_N$ denoting the weight of each vertex. Then there will be **N-1** lines of input containing two space-separated integers **U** and **V** indicating there is an edge between them.

Output

For each case, print one line with "**Case <x>: <y>**", where **x** is the case number and **y** is the vertex chosen as the root of the tree.

Constraints

- **1 ≤ T ≤ 20**
- **1 ≤ N ≤ 30,000**
- **1 ≤ W_i ≤ 10,000**
- **1 ≤ U, V ≤ N & U ≠ V**

Sample Input	Output for Sample Input
1 3 4 5 6 1 2 3 2	Case 1: 3

H Problem H

Update Query Problem

Given two strings **P** and **Q**. You can do **M** number of operations of three types of following formats: each of which is update or query. They will follow the format described below:

- Update: **1 L_1 R_1 C_1**
  You will assign character **C_1** from **L_1** index(1-based) to **R_1** index of string **P**.
- Update: **2 L_2 R_2 C_2**
  You will assign character **C_2** from **L_2** index(1-based) to **R_2** index of string **Q**.
- Query: **3 L_1 R_1 L_2 R_2**
  Take any prefix **X** (possibly empty or the entire substring) from **P[L_1…..R_1]** and any suffix **Y** (possibly empty…) from **Q[L_2…..R_2]** then concat **X** and **Y**. Count the number of distinct strings **X + Y** (X + Y can be empty if an empty prefix and empty suffix is taken) you can get.

Input

First line will contain a single integer representing the number of test cases **T**. For each test case the first line will contain string **P** and the second line will contain string **Q**. Third line will contain a single integer **M** which represents the number of operations. Then each of the **M** lines will contain the above update or query.

Constraints

- **1 ≤ Number of test case ≤ 100**
- **1 ≤ Length(P), Length(Q) ≤ 2 x 10^5**
- **1 ≤ M ≤ 2 x 10^5**
- **1 ≤ L_1 ≤ R_1 ≤ Length(P)**
- **1 ≤ L_2 ≤ R_2 ≤ Length(Q)**
- **Sum of length(P) over all test cases ≤ 2 x 10^5**
- **Sum of length(Q) over all test cases ≤ 2 x 10^5**
- **Sum of M over all test cases ≤ 2 x 10^5**
- **P and Q will contain lower case english letters only.**

Output

For each query you need to print a single integer representing the answer.

Notes

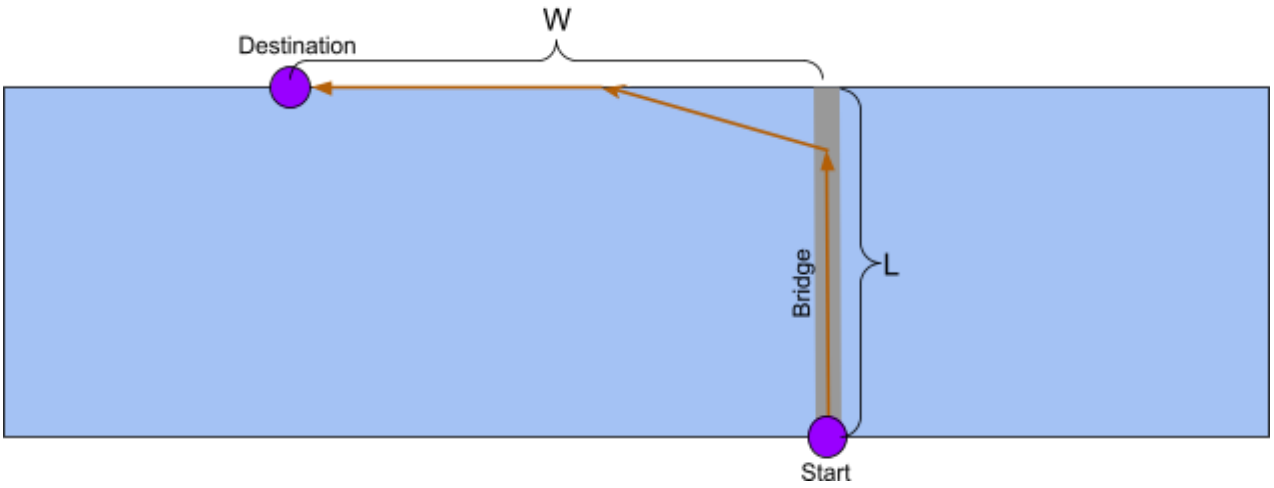
Please use fast input output for this problem.

Sample Input	Output for Sample Input
1 abc de 2 1 1 2 f 3 2 3 1 2	9

I Problem I

Hovercraft

Bangladesh is arguably still a land of rivers but hovercrafts are not frequently seen. In this problem you are requested to find the minimum distance to destination using a hovercraft with certain limitations. The limitation is that it must cover **D** distance on land so that the engine can warm up and then it can cover at most **D** distance in the water. The bridge should also be considered as land.



You will have to cross a river of width **L** starting from the start (As shown in the image) and then reach a destination that is **W** distance away from the other side of the bridge. From the starting location you can cross the river via the bridge and then go to the destination and avoid water completely. In this case your total crossed distance will be **L+W**. Alternatively after crossing the river partially by the bridge you can go through the water and again through land to reach your destination. In this way the total distance covered will be less than **L+W**. Given the value of L and W (Here always **W>L**), your job is to find the minimum distance that needs to be covered to reach the destination from the starting point.

Input

The input file contains at most 1000 lines of input. Each line contains two positive integers that denote the value of **L** and **W (0 < L < W < 1000)**. Input is terminated by a line containing two zeroes which should not be processed.

Output

For each line of input, produce one line of output. This line should contain a floating-point number that denotes the minimum possible distance. This number should have four digits after the decimal point. You can assume that input will be such that small precision errors will not cause any difference in the printed output.

Sample Input	Output for Sample Input
11 23 7 13 0 0	31.2500 18.2500

icpc international collegiate programming contest

ICPC Asia West Regional 2020

ICPC Asia Dhaka Regional Contest

Official Problem Set

14th August 2021

You get 24 Pages, 12 Problems & 240 Minutes

Problem A

The Revenge Of Anti Hash

Given a base **B** and a modulus **M**, the polynomial hash of a string **S**, consisting of only lowercase letters **(a-z)** is defined as below:

```
int GetHash(string str, int B, int M){
      long long hash = 0;
      for (auto chr: str)
            hash = (hash * B + chr - 'a' + 1) % M;

      return hash;
}
```

In other words, first, the letters of the string are replaced by numbers (equivalent to their position in the alphabet, **'a'** gets mapped to **1**, **'b'** to **2, ...** and **'z'** to **26**). This is then considered to be a number in base **B** (the rightmost number is the least significant digit), and the value of this number taken modulo **M** is called the polynomial hash of the string.

Limak the bear loves to hack other contestants in Codeforces. After the recent educational round, he came to know that his friend Swistak used the polynomial hash function stated above to solve the hardest problem! And believe it or not, he was the only one to solve that problem which eventually made him the round champion! Limak is very angry, how can Swistak solve a problem which Limak himself couldn't solve? And worst of all, Swistak used hashing to solve that problem. Limak believes people who use hashing have no real skill, getting 'Accepted' just implies getting lucky, nothing more.

Limak is just a little bear, he is not very good at solving problems. But after hours of scratching his head, he was able to come up with a solution involving *birthday attacks*. That should hack Swistak's solution since educational rounds allow hacking for 24 hours after the round ends. And voila! When he coded it later that night, he was finally able to come up with a case that broke Swistak's solution. And down he goes. From the top place to 153, even below Limak! Limak was overwhelmed with joy.

When Swistak woke up the next morning and casually checked the rank list he was furious. He could not believe what he saw, rolling his eyes in disbelief. He was the only contestant to solve the last problem and that earned him the top place. But alas! No more, because someone hacked his solution and he dropped down more than hundred fifty places. He clicked on the problem to see who hacked him, and his disbelief grew to anger and frustration when he realized it was his friend Limak! He couldn't believe his eyes. "I thought he was my friend, how could he do this to me?", he wondered. He vowed to take revenge. He modified his solution to use double hashing. But just to be extra sure so that Limak can never hack his solution ever again, he hashed

it a few more times resulting in **K** total hashes. Then he submitted his solution which passed the tests and Limak's initial hack as expected.

Afterwards, he rushed to Limak's place and challenged him to a duel. He claimed Limak was jealous and just got lucky while hacking his solution and has no real hacking skills. Feeling overconfident with his new solution, Swistak challenged Limak to hack his new solution and suggested he will retire from competitive programming if Limak can hack his new solution. But if however Limak fails, then Limak must retire instead!

Limak, being provoked like this, takes up the challenge without thinking it through. But he has no clue how to solve it, he is just a little bear after all. He thought about it throughout the whole day but has no idea on how to crack it. With just 4 hours left before the hacking phase ends, he desperately turns to you for help. He knows this isn't exactly fair, but nothing's fair in love and war as they say and he doesn't want to retire from competitive programming. Not now and not ever. Please help Limak solve the following problem and beat Swistak once and for all, thereby saving his career.

Limak will give you **K** pairs of numbers, **(B_1, M_1), (B_2, M_2), … , (B_K, M_K)**. Each pair consists of a base **B** and a modulus **M**. These are the numbers Swistak used to hash strings **K** times in his new solution. Limak needs you to find two **different** strings consisting of lowercase letters only. The strings must have the same hash value when hashed with each of the **K** base/mod pairs with the above described function. Since Codeforces will not accept just any string of arbitrary length as hack inputs, each of the strings also need to be **non-empty** and cannot exceed more than **65536** characters in length. They can be of different lengths though.

Input

The first line contains **T**, denoting the number of test cases. Then **T** test cases follow. The first line of each case contains an integer **K**. The next **K** lines consist of two integers **(B_i, M_i)**. These are the base and mod pairs Swistak used in his hash function.

Constraints

- **$1 \leq T \leq 20$**
- **$1 \leq K \leq 12$**
- **$32 \leq B_i \leq 256$**
- **$1 \leq M_i \leq 256$**

Output

For each test case, output the required two strings separated by a single space. If there is more than one solution satisfying all the above criteria then you may output any of them. You can be assured that there will always be at least one pair of strings.

Sample Input	Output for Sample Input
1 **1** **32 1**	**hello world**

Problem B

Password Shuffling

Working in the IT department of Reynholm Industries is surely hard work. Especially when they have profited 1800 Billion Billion Pounds. Moss is having doubts about the security of the computers, since the employees would just use passwords like their name or “password”. So he decided to generate passwords for everyone. The passwords have these properties:

- They are randomly generated.
- Their length is at least **6** and can be at most **50**. The length is not random.
- The password only contains numerical (0-9), lowercase (a-z) and uppercase (A-Z) letters. Each character of the password is chosen independently and each alphanumeric letter has the same probability to be chosen.

Roy came up with a game to play while they generate the passwords for all employees. He would take a password, split it into multiple groups, shuffle these groups and then concatenate them back together to make a new string. The shuffle is not random, as Roy can shuffle the groups any way he wants. Then he asks Moss to guess how many groups he can split the string into. Moss’ guess would be correct if he can tell the minimum number of groups required to end up with the new string. Can you help him write a program like that?

Input

The first line of the input is an integer **T** (**1 ≤ T ≤ 100)**, denoting the number of test cases. The next **T** lines each contain two strings, $\mathbf{S_1}$ (**6 ≤ $|S_1|$ ≤ 50)** is the original password and $\mathbf{S_2}$ is the password string that is generated after Roy’s shuffle. It is guaranteed that you can always build $\mathbf{S_2}$ from $\mathbf{S_1}$.

Output

For each test, print one line in the format “**Case X: Y**”, where X is the case number and Y is the minimum number of groups the first string needs to be split into so that we can end up with the second string by shuffling the groups.

Sample Input	Output for Sample Input
4 **abcdAs sAdcba** **123abc 123abc** **RUeGz2tjSy0Sbrs3poVA rs3poVARUeGz2tjSy0Sb** **abcd1234 1234abcd**	**Case 1: 6** **Case 2: 1** **Case 3: 2** **Case 4: 2**

Please note that the sample input is for demonstration only, they are not randomly generated. The actual input will be randomly generated.

Sample Explanation

For the third input, the first string can be split into “RUeGz2tjSy0Sb” and “rs3poVA”, which would result in the second string after changing the order and merging them together.

Problem C

Chip's Challenge

Chip's Challenge is a classic top down puzzle game played on a **N x M** grid. Released in 1989, the protagonist of the game is the high-school nerd Chip McCallahan, who has met Melinda The Mental Marvel in the school science laboratory and fell in love at first sight! Chip must navigate through Melinda's clubhouse, consisting of several increasingly difficult puzzles, in order to prove himself and acquire Melinda's heart and also gain access to the very exclusive Bit Busters Club. Such is the premise of this vintage game, cliched, but who cares about the plot when the gameplay is so cool?

The game consists of more than one hundred two-dimensional levels. Gameplay involves using the arrow keys of the numeric keyboard or mouse to move Chip around each of the levels in turn. To progress through each level, Chip needs to interact with various game elements like computer chips, buttons, locked doors, water and lethal monsters. There are also some levels containing block-pushing puzzles and dodging enemies! Chip can move on to the next level by collecting enough chips to open the chip socket at the end of each level and get to the exit. Most levels have a time limit and levels can also be skipped by entering an appropriate four letter password.

Below is an example of a level from the actual game.

In this problem, we will attempt to solve a simplified variation of the game Chip's Challenge. You will be given a **N x M** grid with Chip's starting position and also the position of Melinda The Mental Marvel. The grid will consist of empty cells, water, lethal monsters and locked doors only. The rules of this game are a bit different than the original version, so pay close attention to the following details. Chip starts from his initial position and wants to travel to Melinda's location. To do so, he can navigate to any four of the adjacent cells from his current position using the arrow keys up, down, left and right. Provided the cell he wants to move to is empty and inside the grid, he will move to the cell and the whole process takes **exactly 1 second**. Otherwise, Chip is not allowed to make a move.

He can also click on any cell that is not empty and attempt to destroy it. Any cell that is not empty can be destroyed and once destroyed, the cell becomes empty for the rest of the game. Note that the cell does not necessarily have to be adjacent to Chip's current position while destroying it. Initially empty cells are marked using the character **'.'** (dot) and both Chip's starting position and Melinda's positions are initially empty cells and are different. Every non-empty cell has a power value and destroying a cell with power value **X** takes exactly **X** clicks, hence **X** seconds in total. That is, per click it takes 1 second. Keep in mind that the non empty cells can either consist of water, monsters or locked doors.

Cells containing water are marked using lowercase letters, therefore there can be **26** different types of water. The power value of a cell containing water is equivalent to its position in the alphabet. That means if a cell contains the character **'a'**, it's power value is **1**, if a cell contains the character **'b'** it's power value is **2**, and similarly for **'z'** it's power value is **26**. Monsters are marked with uppercase letters, and their power values are calculated by adding **26** with its position in the alphabet. So if a cell contains **'A'**, the power value of it will be **27**, for **'B'** it will be **28** and similarly for **'Z'** it will be **52**. Lastly, cells containing locked doors will be represented by the digits **0-9**. The power value of a cell containing a locked door will be the value of the digit plus **53**. If a cell contains a locked door labeled **'0'**, it's value will be **53**, for a cell having locked door labeled **'1'**, the power value will be **54** and for the locked door **'9'**, the power value will be **62**.

Chip can't wait to rejoin the love of his life and join the Bit Busters club with her and he wants to reach her in the shortest time possible. Given the description of the maze, the initial position of Chip McCallahan and Melinda The Mental Marvel, can you suggest how to play Chip's Challenge optimally so that Chip can reach Melinda as fast as possible?

Input

The first line will contain a single integer **T** and then **T** test cases follow. Every test case contains two lines. The first line contains 6 integers - **N, M, C_X, C_Y, M_X, M_Y**. The grid dimensions are denoted by **N** and **M**, **(C_X, C_Y)** indicates the starting position of Chip and **(M_X, M_Y)** the starting position of Melinda. The maze will be generated pseudorandomly. The second line of every case contains one integer called **seed**, and the maze is generated by following the C++ code snippet described below. Note that seed is a global variable that is initialized once at the start of every test case with the value given, and gets updated after each **update_seed()** call.

```
long long seed;

char maze[2021][2021];

void update_seed(){
    seed = (seed * 1000003 + 10007) % 1000000009;
}

void gen_maze(int N, int M, int Cx, int Cy, int Mx, int My){
    for (int i = 1; i <= N; ++i){
        for (int j = 1; j <= M; ++j){
            if ((i==Cx && j==Cy) || (i==Mx && j==My)){
                maze[i][j] = '.';
                continue;
            }

            update_seed();
            int power = seed % 63;

            if (power == 0)
                maze[i][j] = '.';

            else if (power <= 26)
                maze[i][j] = power - 1 + 'a';

            else if (power <= 52)
                maze[i][j] = power - 27 + 'A';

            else
                maze[i][j] = power - 53 + '0';
        }
    }
}
```

Constraints

- **$1 \le T \le 10$**
- **$1 \le N, M \le 1000$**
- **$1 \le C_X, M_X \le N$**
- **$1 \le C_Y, M_Y \le M$**
- **$(C_X, C_Y) \ne (M_X, M_Y)$**
- **$0 \le seed \le 1000003$**

Output

Print the case number followed by the shortest time to complete Chip's Challenge if played optimally. Check the sample for more details.

Sample Input	Output for Sample Input
2 3 3 1 1 3 3 3 1 3 1 1 1 3 1	Case 1: 29 Case 2: 59

Explanation

For the first test case, the maze looks like:

.bl

JaS

Gv.

The shortest path from (1,1) -> (3,3) is the following:

- Destroy cell (1, 2) in **2** seconds
- Destroy cell (2, 2) in **1** second
- Move right from cell (1, 1) to cell (1, 2) in **1** second
- Move down from cell (1, 2) to cell (2, 2) in **1** second
- Destroy cell (3, 2) in **22** seconds
- Move down from cell (2, 2) to cell (3, 2) in **1** second
- Move right from cell (3, 2) to cell (3, 3) in **1** second
- Therefore the overall journey takes **2 + 1 + 1 + 1 + 22 + 1 + 1 = 29 seconds**

Problem D

Constructing Strings

Let's define a special string **S** which follows the following conditions :

1. **S** has length **N**
2. **S** consists of lowercase English letters only
3. It satisfies **Q** conditions of the following format :
   L R : Substring of **S** starting from **L** to **R** (both inclusive) must be palindrome (1-based indexing).

You have to count the number of **distinct special** strings. As the answer can be large, print it modulo **998244353.**

Input

The first line will contain a single integer **T(1 ≤ T ≤ 10)**, denoting the number of test cases.

The first line of each test case will contain 2 space separated integers : **N Q (1 ≤ N, Q ≤ 10^5).**

The next **Q** lines of each test case will contain 2 space separated integers : **L R (1 ≤ L ≤ R ≤ N)**, describing a condition.

Output

For each test case, first print the case number and then print the number of valid strings for that test case modulo **998244353**.

Sample Input	Output for Sample Input
2 2 1 1 2 12 4 1 3 10 11 1 5 4 6	Case 1: 26 Case 2: 45855352

Problem E

The Lieutenant's Exam

Sergeant Terry Jeffords is attending an exam where he has to answer only one MCQ question. There are **N** options for this question. He has to pick exactly one option as his answer. Some of them are correct and the rest are wrong. So if he picks any of the correct options, he will pass the exam. But unfortunately, Terry doesn't know which options are correct.

These **N** options for the question are actually **N** distinct binary strings. Terry suddenly notices that the teacher in the exam room is writing a binary string on the whiteboard. He suspects that the answer to the question is a subsequence of the string that is being written by the teacher.

The teacher is writing an infinitely long binary string. She starts with an empty string. Then in each step, she writes a digit or removes the last written digit. This is how she decides what to do in each step:

If this is not the first step and she has not removed a digit in the previous step:

i) She writes "0" with **X** probability

ii) She writes "1" with **Y** probability

iii) She removes the last written digit with (**1-X-Y**) probability.

Otherwise,

i) She writes "0" with $\frac{X}{(X+Y)}$ probability

ii) She writes "1" with $\frac{Y}{(X+Y)}$ probability

So the teacher never removes digits in two consecutive steps.

If after any step, the currently written string on the whiteboard contains any of the **N** options as a subsequence, Terry would pick it as the answer and end the exam. If there are multiple options that are subsequences of the current string, Terry will pick the option which has come first in the input options.

What is the probability of Terry picking a correct answer?

Input

The first line will contain a single integer **T** denoting the number of test cases. Each test case will have an integer **N** and two numbers **X** and **Y** in the first line. Next line will contain **N** space separated binary strings. The last line of the test case will contain a binary string of length **N.** If the **i^{th}** bit of this string is "1", then the **i^{th}** option is correct and if the **i^{th}** bit of this string is "0", then the **i^{th}** option is incorrect. It is guaranteed that there is at least one correct option.

Constraints

- **1 ≤ T ≤ 100**
- **1 ≤ N ≤ 5**
- **1 ≤ Length of the binary string in an option ≤ 8**
- **0 < X, Y**
- **(X+Y) < 1**
- **X and Y will have at most two digits after the decimal point**

Output

For each case, print the case number and the answer in a single line. Please see the sample for details. Your answer will be considered correct if its absolute error doesn't exceed **10^{-6}.**

Formally: If your answer is a, and the answer of the jury is b, the checker program will consider your answer to be correct, if $|a - b| \leq 10^{-6}$.

Sample Input	Output for Sample Input
5 2 0.3 0.45 1 0 10 2 0.57 0.25 10 0 10 2 0.57 0.25 0 10 01 5 0.74 0.16 10 00001 10111011 11 0101000 11100 5 0.46 0.14 00111111 11010000 01101110 10011000 10011010 10111	Case 1: 0.59999999999999997 Case 2: 0.26451612903225806453 Case 3: 0.00000000000001000000 Case 4: 0.90347426984368913979 Case 5: 0.72854717043475962230

Problem F

Fair Set

Given a list of numbers, Hermione wants to choose a collection of **K** numbers with minimum unfairness. Unfairness of a collection **A** is defined as $\sum_{i=1}^{|A|} \sum_{j=i+1}^{|A|} (A_i - A_j)^2$. You need to find out the value of the minimum unfairness and also the number of such collections with minimum unfairness. A collection **S** is different from another collection **T** if there exists at least one number whose frequency is different in **S** and **T**.

Input

The first line of input contains a single integer, **T (T ≤ 500)**. Then **T** test cases follow. Each of these **T** test cases starts with two integers **N (1 ≤ N ≤ 10,000) and K (1 ≤ K ≤ F).** Then **N** lines follow, where each line contains two integers, **v (1 ≤ v ≤ 10,000)** and **f (1 ≤ f ≤ 100),** which indicates the number **v** is present **f** times in the list of numbers. **F** is the summation of all **f**, i.e. the size of the list of numbers. The sum of **N** for all cases is less than **2,000,000**.

Output

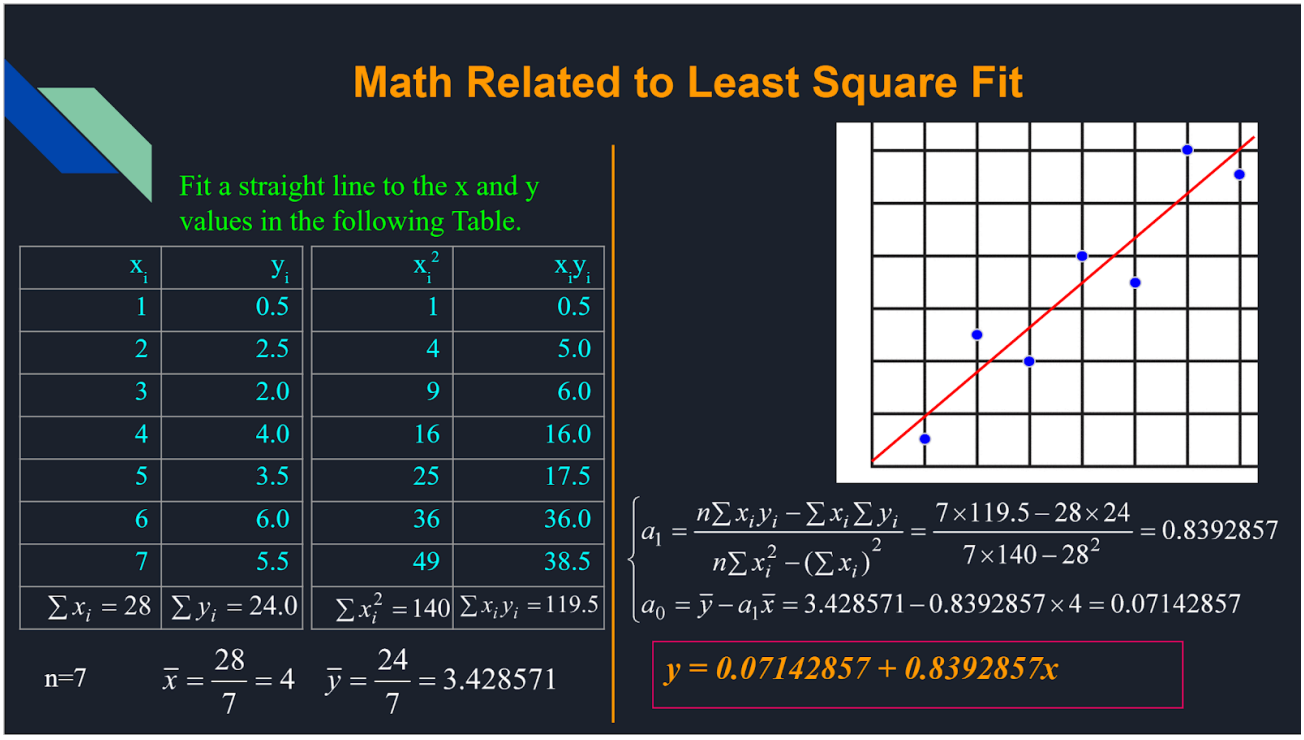
For each of the cases, output "**Case <x>: <y> <z>**" in a separate line, where **x** is the case number, **y** is the minimum possible unfairness of a collection of **K** numbers, and **z** is the number of such collections.

Sample Input	Output for Sample Input
2 5 3 3 1 5 1 6 1 1 1 9 1 4 2 234 2 1 1 5 1 3 1	Case 1: 14 1 Case 2: 0 1

Problem G

Creating Assignment

Tomisu teaches a course titled “Numerical Analysis” in Southeast University of Bangladesh. During Covid-19 pandemic he developed a practice to take online exams of students. He has given math to find the best fit line using least square method in linear regression analysis. The slides that he uses to teach in the class goes something like this:



x_i	y_i	x_i^2	$x_i y_i$
1	0.5	1	0.5
2	2.5	4	5.0
3	2.0	9	6.0
4	4.0	16	16.0
5	3.5	25	17.5
6	6.0	36	36.0
7	5.5	49	38.5
$\sum x_i = 28$	$\sum y_i = 24.0$	$\sum x_i^2 = 140$	$\sum x_i y_i = 119.5$

As it is an online exam, he gives different maths to different students. But after sending different PDF questions to all students he realized that he has made a mistake: He told his students that he will give all maths in such a way so that the regression line goes through the origin (0,0) (the value of a_0 is always zero). In this way the students will be able to catch calculation errors in almost all cases. But the value of a_1 will be different as points given in each math are different (So a_1 is the significant answer). But somehow he forgot to ensure that the value of a_0 is always zero. So for now for all math he has to add one point that will make the regression line go through the origin but the slope will be unchanged (So that if someone solves the math before getting the correction, his significant answer does not change).

Input

The input file contains at most **1000** sets of inputs.

Each set starts with an integer **N (3 ≤ N ≤ 1000)** which denotes the number of points. Each of the next **N** lines contain two floating-point numbers which denote the Cartesian coordinate of a point. You can assume that all

the coordinate values in the input will be non-negative and will not exceed **1000**. **You can also assume that the regression line for these given points will not go through the origin.**

Input is terminated by a single zero.

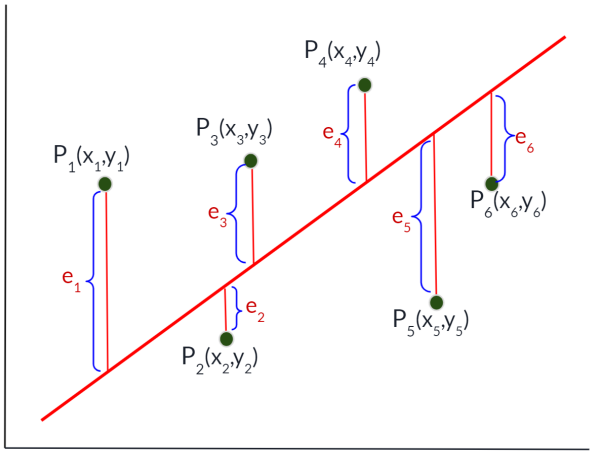
Output

For each case of input produce one line of output. This line contains the case number that is followed by two floating-point numbers. These two numbers denote the Cartesian coordinate of the newly added point that will ensure that the regression line passes through the origin. All these numbers should be rounded to exactly three digits after the decimal point. **You can assume that the inputs will be such that small changes in output values will not change the printed value. Look at the output for sample input for details.**

Sample Input	Output for Sample Input
3 1.0000 1.0000 2.0000 4.0000 5.0000 8.0000 7 1.0000 0.5000 2.0000 2.5000 3.0000 2.0000 4.0000 4.0000 5.0000 3.5000 6.0000 6.0000 7.0000 5.5000 0	Case 1: 2.667 4.641 Case 2: 4.000 2.857

Linear Regression (Least Square Method):

We will try to cover some basics of least-square regression here using the figure on the right just in case it helps. There are six points $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$, $P_4(x_4, y_4)$, $P_5(x_5, y_5)$ and $P_6(x_6, y_6)$ on the right and suppose the long red line from left to right is a candidate for the regression line found using least square fit. The error for any point is the distance of it from the candidate regression line along the y-axis. So for points P_1, P_2, P_3, P_4, P_5 and P_6 the errors are e_1, e_2, e_3, e_4, e_5 and e_6 respectively. And if the red line is the regression line using least square linear regression then $e_1^2+e_2^2+e_3^2+e_4^2+e_5^2+e_6^2$ is minimum. If the equation of this regression line is $y = a_0 + a_1x$ then the value of a0 and a1 are found using the formulas on the right and they are derived using partial derivatives.



$$\begin{cases} a_1 = \dfrac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2} \\ a_0 = \bar{y} - a_1 \bar{x} \end{cases}$$

Problem H

XOR

You are given a sequence of **n** numbers **A** and **q** queries. Each query consists of two indices **l** and **r (1** $\leq$ **l** $\leq$ **r** $\leq$ **n)**. Find the value of the maximum subsegment XOR inside the subsegment **A[l...r].**

Note,
A subsegment is a contiguous sequence which is part of the original sequence.
Subsegment XOR is the XOR of all the numbers of the subsegment.

Input

First line of the input contains **T**, the number of the test cases. T test cases follow. Each case starts with 2 integers n and q. Next line contains n integers describing the sequence A. Next q lines contain 2 integers each: l and r.

Constraints

$1 \leq T \leq 10$
Sum of n over all test cases ≤ 100000
Sum of q over all test cases ≤ 100000
$0 \leq$ Each number of the input sequence $\leq 2^{20} - 1$

Output

For each case, print the case number. Then print the answer for each query in separate lines. See the sample input output for more details.

Sample Input	Output for Sample Input
1 5 2 1 0 1 0 0 1 3 4 5	Case 1: 1 0

Problem I

Hitmen

In order to save humankind, **n** potential criminals need to be taken down. The criminals are numbered from **1** to **n**. The government of Digital Utopia(DU) has announced different prize money for taking down each of them. The International Crime Prevention Corporation(ICPC) is planning to hire a bunch of hitmen to take down the criminals. There are **m** hitmen, numbered from **1** to **m**. But every hitman cannot take down every criminal. In fact, for the i^{th} criminal, there exists a range $[L_i, R_i]$ such that the hitmen whose ids are between L_i and R_i, can only take down that criminal. As expected, hiring a hitman will be costly. But once a hitman is hired, he will be obliged to take down as many criminals as ordered without any additional charge.

The chief recruiting officer of ICPC has hired a freelancer to decide which hitmen to buy and which criminals to take down so that their profit is maximized. While calculating the profit, prize money gained from taking down criminals should be added and expenses for hiring hitmen should be subtracted.

While solving the problem, the freelancer has discovered a surprising fact about the ranges of hitmen for the criminals: the given ranges never partially overlap, but one can be completely inside another range. Even after this discovery, the freelancer was unable to solve the problem. Can you help him to solve it?

Input

First line of input will contain an integer **T**, denoting the number of test cases. Each test case will start with a line containing **n** and **m.** Each of the next **n** lines will contain **3** space-separated integers denoting the prize money for taking down a criminal and the lowest and highest id of the hitmen that can take down that criminal. The last line for a test case contains **m** space-separated integers, denoting the hiring costs of the hitmen in the order of their id.

Output

For each test case, print the case number and the maximum attainable profit. Check the sample i/o for details.

Constraints

$1 \leq T \leq 100$

$1 \leq n, m \leq 50,000$

$0 \leq$ prize money for taking down a criminal $\leq 10^9$

$0 \leq$ cost of hiring a hitman $\leq 10^9$

$1 \leq L_i \leq R_i \leq m$

$2 \leq \Sigma n + \Sigma m$ over all test cases $\leq 5 \times 10^5$

For any $1 \leq i, j \leq n$ and $i \neq j$, the following condition will never be true:

$L_i < L_j \leq R_i < R_j$

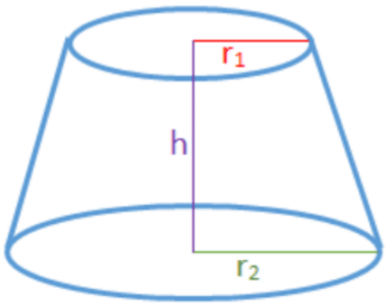
In other words, no two ranges will partially overlap.

Sample Input	Output for Sample Input
2 3 3 10 1 1 5 2 2 8 3 3 1 1 1 2 3 41 2 3 78 3 3 45 81 27	Case 1: 20 Case 2: 92

Problem J

Ant on the Conic

A teacher always gives easy exercises in the classroom but tougher in exams. Legend says, there was a student in the class who was an expert in mathematics. The student was way ahead of others in the class. Most of the others used to memorize the entire math book, while the student we are talking about never memorized, always solved math methodically. However, once before one exam the student did not understand one tough math from the book and thought: “Okay what’s the probability that this one will appear in the exam paper out of hundreds of the problems!”. And guess what exactly that one appeared. While he could not solve it, others in the class could solve it very easily!! Anyway the student is now the teacher. It’s now the payback time, but in his own way! In the class the students were given an easy problem: “Given two points on the surface of the earth, find the shortest distance between them and find the maximum latitude it will reach by a shortest path”. But in the exam, the following problem was given: “Given two points on a hill, find the shortest distance between them and also find the maximum height one can reach using a shortest path.” Let’s formalize the problem:



Given a cone-like solid shape as in the picture. The imaginary line going through the centers are perpendicular to both of the circular faces of the shape. The height is h, the radii of the two faces are r1 and r2. Two points are given on the slanted body. You need to find the shortest distance from one point to the other which wraps around the conic body at least n times (note this wrapping does not need to happen at the same height level). You will also need to find the maximum height that you can reach by one of the shortest paths. You can neither go through the circular surfaces nor go through the solid shape. You always have to be on the slanted body (it’s allowed to touch or go through the border of the slanted body.

Wrapping around the body is a vague definition. If you are familiar with the so-called winding number, you can think of it like a winding number around the vertical line through the centers of the circles. Another way to imagine it would be, to imagine yourself as a point on the vertical line through the centers. Now keep looking at the path one traverses from one point to the other. Once the path traversal finishes, count how many times you fully rotated around the vertical line. Please note, rotating 30 degree clockwise and then 390 degree counterclockwise means you rotated only 360 degree counterclockwise hence only one rotation. If the path ever comes to the vertical line (this may happen if one of the radii is zero) then you can consider the path wrapped around the vertical line (or the body) infinite times.

Input

First line of the input contains the number of tests, T. Hence T cases follow. Each case consists of 4 lines. First line describes the cone using three integers: h r_1 r_2 where h is the height, r_1 and r_2 are the radii of the upper and lower circles respectively. Second and third lines each describe a point using two integers: z theta. Here z is the height from the base and theta is an angle similar to longitude. That is, the described point is at z height from the base, and at theta angle around the vertical line through the centers of the circles (for a fixed 0 angle). The fourth line contains an integer n, the number of wrapping around the 3D body.

Constraint
$T \leq 15{,}000$
$1 \leq h \leq 10$
$0 \leq r_1, r_2 \leq 10$
$0 < r_1 + r_2$
$0 \leq z \leq h$
$0 \leq theta < 360$
$0 \leq n \leq 10$

Output

For each test case, output the case number followed by two floating-point numbers: the first one is the shortest distance and the second one is the maximum height it can reach. If the numbers are within 10^{-3} of the correct answer (by an absolute or relative) it will be accepted.

Sample Input	Output for Sample Input
2 10 0 10 0 0 5 0 0 10 0 1 0 0 9 0 10	Case 1: 7.071067811865 5.000000000000 Case 2: 11.054863183233 10.000000000000

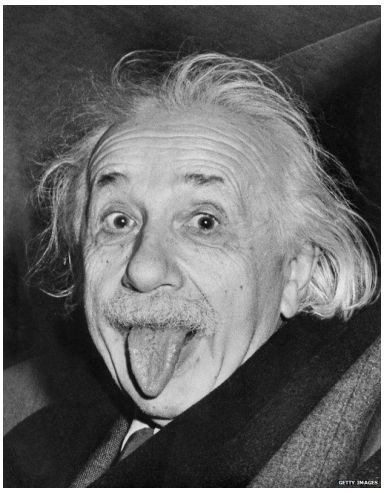
Explanation

I am a very good and student-friendly teacher, hence I gave you the second case!

Problem K

Fare Thee Well, My Friend

Fare thee well,
my friend.
Fare thee well.
Keep this torrid land
in your heart,
wherever you are going.
May you find peace
and happiness within.
- Albert Einstein



Input

There is no input for this problem.

Output

In a single line, print the number of words on the poem (excluding the name of the poet) written above.

Sample Input	Sample Output
	28

Note:
The sample output is just to demonstrate the format, it may not be the actual word count of the poem.

Problem L

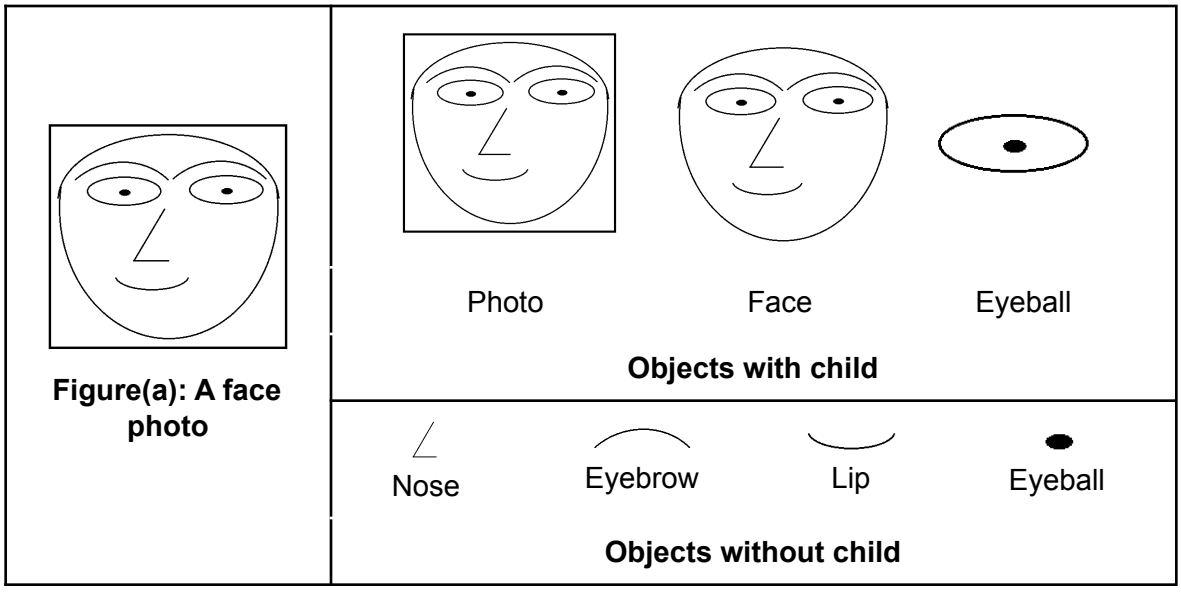
Object Detection

Detecting objects within an image is a very common research problem. An image can be represented as an $(N * M)$ matrix where each cell has a color intensity represented by character **'a'** to **'z'**, **'.'** and **'#'**. In an image, cells are considered as pixels. Pixel value **'.'** means it's blank or zero pixel (no ink). Otherwise, it's a non-zero pixel.

The following characteristics define an object within an image.

- Any two non-zero pixels sharing at least one common corner in the grid are considered to be part of the same object.
- An object can have multiple separate blank spaces inside it (see samples). Everything inside those blank spaces (if any) are part of that object.
- One or more objects may sit inside the blank space of another object. The outer object is called the parent. The immediate inner objects are called the child objects. Child objects of the same parent are called siblings.

To get a clear concept of parent, child and sibling relationships amongst objects, see the following pictorial example.



Figure(a): A face photo

The given Photo object has one child object named Face. The Face object has six child objects which are: two Eyebrow objects, two Eye objects, one Nose object and one Lip object. So, these six objects are siblings to each other. Each Eye object contains an Eyeball object as a child object. The Eyeball object of the left Eye and the Eyeball object of the right Eye are not siblings.

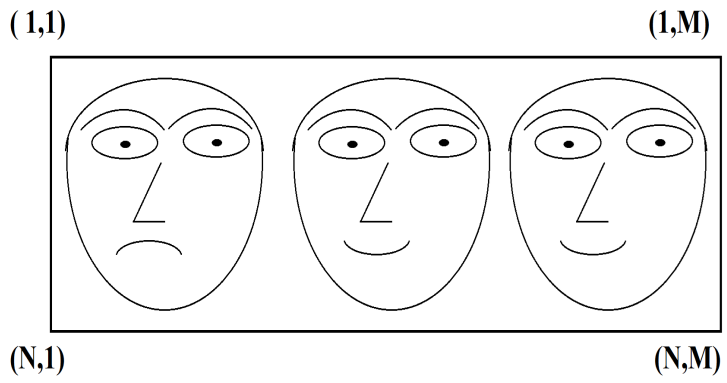
Two objects A and B are identical if all the following conditions are satisfied.

- They are siblings.
- Object A can be achieved after translating the axis of object B.
- All of the corresponding pixel **intensity** of A and B are equal, including pixels from their child objects.

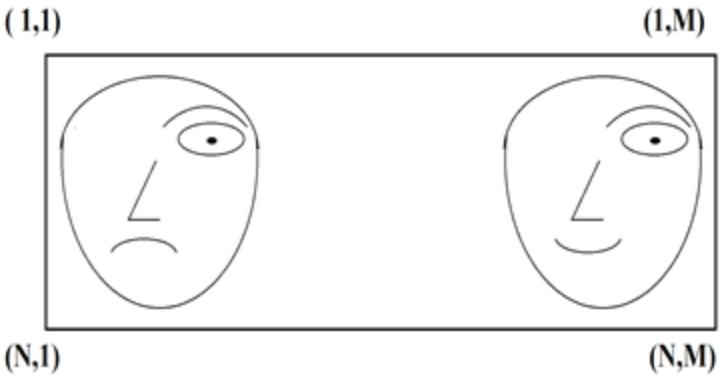
In the previous figure, the two Eyebrow objects are identical, so are the two Eye objects. As the Eyeball objects are not siblings, they are not considered to be identical.

In this problem, you have to identify all the distinct objects in an image and print their rightmost upper pixel position. The rightmost upper pixel position of an object is the cell (x,y) belonging to the object such that y is maximized, and then x is minimized. If the object is a parent, then you have to print all the child objects based on their rightmost upper pixel position. The child objects must be printed in order of decreasing y of the rightmost upper pixel. If y of the rightmost upper pixel of two objects are the same, then print the one having smaller x first.

An illustration is provided in the figure (b). In figure (b) we can see it has three faces numbered as A, B and C (from left to right). Face B and C are identical. So we output Face C by comparing the rightmost upper pixel of B and the rightmost upper pixel of C. Also, the Eyes and Eyebrows inside each face are identical.



Figure(b): Photo consists of three faces ***A, B*** and ***C*** (from ***left to right***)



Figure(c): Output image derived from above figure.

Distinct objects of Figure(b) in a tree structure according to parent-child relation:

```
|---Photo Frame
    |---Face C Border
        |---Eyebrow of C
        |---Eye of C
            |---Eyeball of C
        |---Lip of C
        |---Nose of C
    |---Face A Border
        |---Eyebrow of A
        |---Eye of A
            |---Eyeball of A
        |---Lip of A
        |---Nose of A
```

Given an image as the input, you will need to output the tree structure. See the Sample I/O for understanding.

Input

The first line of the input contains an integer **T(1≤T≤100)**, denoting the number of test cases. Each of the test cases starts with two integers, **N, M (3≤N, M≤1000)** denoting the size of the image. Next, there will be **N** lines of input. The **y-th** character of the **x-th** line represents the cell **(x,y)**. You may safely assume first row, first column, last row and last column of the input image has character **'#'** representing the photo frame of the image and no object touches the image frame. Other characters of the input are **'a'** to **'z'** and **'.'**. Character **'.'** represents a blank cell and character from **'a'** to **'z'** represents pixel intensity. **Note that, in a test file** $\sum(N \times M) \leq 2 \times 10^7$**.**

Output

For each test case, print the case number in the form of **"`Case K:`"** in a single line, where **K** is the case number. From the next line, print each of the distinct objects in the form of **"`|---(x,y)`"** in a single line where **(x,y)** stands for the rightmost upper pixel position. If the object is composed of one or more child objects, then also print them from the next line adding four leading spaces.

Note that distinct objects will be printed based on their rightmost upper pixel positions. You should not print any trailing space. See the sample I/O for more clarity.

Sample Input

```
3
20 35
###################################
#.................................#
#....oooooooooooooooooooooooooo...#
#...o........................o..#
#..o...ooooooo......ooooooo....o..#
#..o..x.......x....x.......x...o..#
#..o....ooo............ooo.....o..#
#..o...o...o..........o...o....o..#
#..o...o.x.o....l.....o.x.o....o..#
#..o...o...o....l.....o...o...o...#
#..o....ooo.....l......ooo...o....#
#...o...........l...........o.....#
#....o.......jjjj..........o......#
#.....o...o........o......o.......#
#......o...oo....oo......o........#
#.......o....oooo.......o.........#
#........o............o..........#
#.........voooooooooooov..........#
#.................................#
###################################
15 17
##################
#................#
#.aaaaa..aaaaa..#
#.a...a..a...a..#
#.a.b.a..a.c.a..#
#.a...a..a...a..#
#.aaaaa..aaaaa..#
#................#
#.abc....aaaaa..#
#........a...a..#
#.a..c...aaaaa..#
#.b..b...a...a..#
#.c..a...aaaaa..#
#................#
##################
13 24
########################
#......................#
#.bbbbbbbbb..bbbbbbbbb.#
#.b...b...b..b...b...b.#
#.b...b.a.b..b...b.a.b.#
#.b...b...b..b...b...b.#
#.bbbbbbbbb..bbbbbbbbb.#
#.b...b...b..b...b...b.#
#.b.a.b.a.b..b.a.b.a.b.#
#.b...b...b..b...b...b.#
#.bbbbbbbbb..bbbbbbbbb.#
#......................#
########################
```

Output for Sample Input

```
Case 1:
|---(1,35)
    |---(4,32)
        |---(6,28)
        |---(8,27)
            |---(9,25)
        |---(14,20)
        |---(9,17)
Case 2:
|---(1,17)
    |---(3,14)
        |---(5,12)
    |---(9,14)
    |---(3,7)
        |---(5,5)
    |---(11,6)
    |---(9,5)
    |---(11,3)
Case 3:
|---(1,24)
    |---(3,22)
        |---(5,20)
```

Explanation

In the third sample case, the ‘a’s in (5, 9), (9, 5) and (9, 9) are siblings with each other because they are inside the blank spaces of the same parent. Similarly, the ‘a’s in (5, 20), (9, 16) and (9, 20) are also siblings with each other.

Problem A

The One With Infinite Squares

You are given **N** two-dimensional points $P_1, P_2, P_3, \ldots\ldots, P_N$. These points are **randomly generated one by one**. The coordinates of the points are non-negative and less than 10^5. It is possible that two different points have the exact same coordinates.

After the i-th point is generated, a square can be drawn in many different ways so that it contains all of the points that are generated till now. The side length and the orientation of the square can be anything. We consider that a square contains a point if and only if the point lies inside the square or on its boundary.

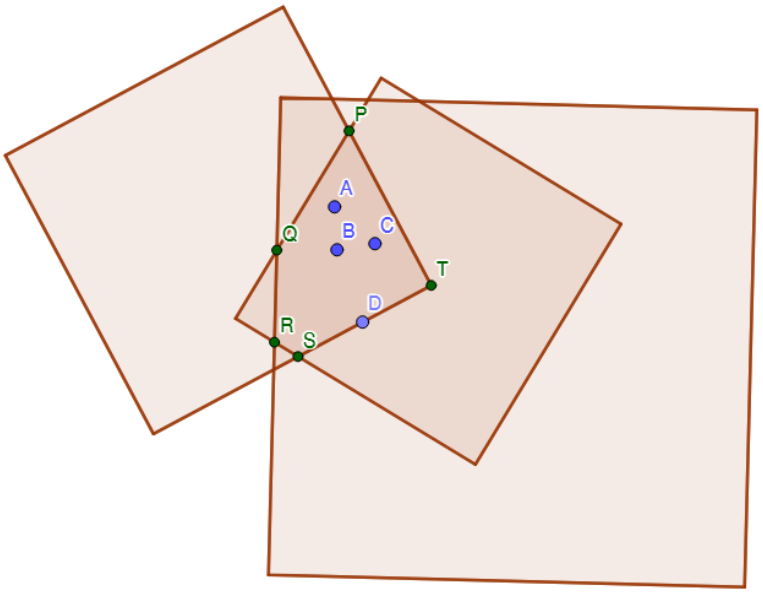
Let S_i be the set of all possible such squares which can individually contain all of the points from P_1 to P_i.

Let, the strength of a set **Q** containing some squares be:

Strength(Q) = 2 x (area of intersection of all squares in **Q**)

You need to determine the following sum:

$$\text{Sum} = \sum \text{Strength}(S_i) \quad (\text{where } 1 \le i \le N)$$



In the figure above, we want to cover the blue points (A, B, C, D) with any square. Each of the three squares drawn above contains all of the points individually. The strength of a set containing these three squares will be (2 x area of polygon PQRST). Note that for the given blue points, there are many squares other than these three, that can contain them individually.

Input

The first line will contain a single integer **T** denoting the number of test cases. Each test case will have an integer **N** and two integers **seed1** and **seed2**. You have to generate the points using **seed1** and **seed2** using the following pseudocode for random generator:

Random Generator Pseudocode:

```
global variables
    seed1, global var1
    seed2, global var2
end global variables

procedure GET-RANDOM()
  seed1 = (seed1 * 1103515243 + 12345) mod (2012345671)
  seed2 = (seed2 * 1092104927 + 54321) mod (2094828103)
  r = (seed1 * seed2) mod 100000
  return r
end procedure
```

Here you can find a sample C code for generating and printing **n** random points for **t** test cases. The following code is given so that you can better understand how to generate the **n** points using the seeds.

Sample C code for generating and printing n points :

```
#include<stdio.h>
long long int seed1, seed2;
int get_random() {
    seed1 = (seed1 * 1103515243 + 12345)%(2012345671);
    seed2 = (seed2 * 1092104927 + 54321)%(2094828103);
    int r = (seed1 * seed2) % 100000;
    return r;
}
int main(){
    int t, cs, i, n;
    scanf("%d",&t);
    for(cs = 1; cs<=t; cs++){
        scanf("%d %lld %lld",&n,&seed1,&seed2);
        for(i = 1; i<=n; i++){
            int x = get_random();
            int y = get_random();
            printf("%d %d\n",x,y);
        }
    }
    return 0;
}
```

Constraints

- **$1 \le T \le 10^3$**
- **$1 \le N \le 2 \times 10^7$**
- **$\sum N$ over all test cases $\le 2 \times 10^7$**
- **$1 \le seed1, seed2 \le 10^9$**

Output

For each case, print the case number and the value of **"Sum"** as described in the problem.

Please see the samples for more details.

Sample Input	Output for Sample Input
2 4 1 3 100000 842 91	Case 1: 5471244680 Case 2: 1997153783304545

Note About Sample

For the first test case in the sample, the 4 generated points are:

- (80412, 70783)
- (29920, 9845)
- (19576, 38405)
- (83584, 94275)

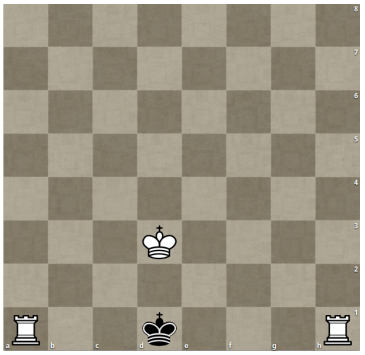
Problem B

Kings Vs Rooks

Chess is a two-player strategy board game played on a **8 x 8** board. Each player begins with **16** pieces where each type of piece moves differently. Player 1 has all the white pieces and player 2 all the black ones, with white being the one to move first and then players take turns alternatingly until the game ends. The game is played on a square board of eight rows and eight columns. The rows are called ranks and are numbered 1 to 8 from bottom to top from white's perspective. Similarly, the columns are called files and are denoted from a to h from left to right according to white's perspective. The objective of the game is to threaten the opponent's king in such a way that escape is not possible, i.e, checkmate the opponent. If a player's king is threatened, it is said to be in check and the player must remove the king from check on the next move by either moving to a safe square or capturing or blocking the attacking piece(s). The figure shows the starting position for the game of chess.

In this problem, we'll be considering a slightly modified version of chess with the following rules:

1. The game will be played on a **N x N** board
2. The game will consist of exactly 4 pieces - The white king, the black king, and two white rooks. All four pieces must be placed on different squares of the board.
3. Kings can move to any adjacent square (horizontally, vertically, or diagonally), unless the square is outside the board or occupied by a friendly piece or if the move would place the king under check.
4. Rooks can move to any square in the same rank or file, but not by jumping over other pieces. Also it cannot land on a square occupied by a friendly piece.
5. A piece **X** is threatened by another enemy piece **Y,** iff **Y** can reach **X** in exactly one legal move.
6. A piece **X** can capture another enemy piece **Y**, iff **X** can reach **Y** in exactly one legal move. On capturing, piece **Y** is removed from the board and **X** takes its place. Note that if **X** is the king, then it cannot capture **Y** if **Y** is guarded by an enemy piece. That is the king can never move and put itself in danger where it is attacked by an enemy piece.
7. The two kings cannot ever be in adjacent (horizontally, vertically or diagonally) squares.
8. The objective of the game is to place the four pieces on unique squares of the board in such a way that if it is black's move, then the black king is attacked and has no legal moves. That is, the black king must be threatened by at least one of the white rooks and has no way to move to a safe square or eliminate the checks (by capturing the attacking pieces if not guarded). Note that it is not necessary for the board configuration to be legal following the exact rules of chess. That is, if **N = 8,** there might be valid configurations in our modified game which might never appear in a game of chess. Like the figure shown beside, which can never occur in a proper game of chess given that it is black's turn to move, although it would be a perfectly valid configuration in our game.



Given **N**, you need to count how many ways you can place one white king, one black king, and two white rooks in the chessboard such that the black king is threatened and has no way to escape the threat given it is black's turn to move. Two configurations are different if either any of the squares contain different pieces or one square contains a piece but the other does not. Note that every square is uniquely represented by it's rank and file and hence mirrored or reversed configurations of the same configuration will be considered as different configurations. Also note that the white king, black king and white rooks are distinguishable but the two white rooks are identical. So in the figure above, if we swap the two rooks, the new configuration will be the same as the old one.

Since the number of ways can be huge, you need to output it modulo **7340033**.

Input

The first line contains the number of test cases denoted by **T (1 ≤ T ≤ 20)**. The next **T** lines contain the board dimension **N (1 ≤ N ≤ 100000)**.

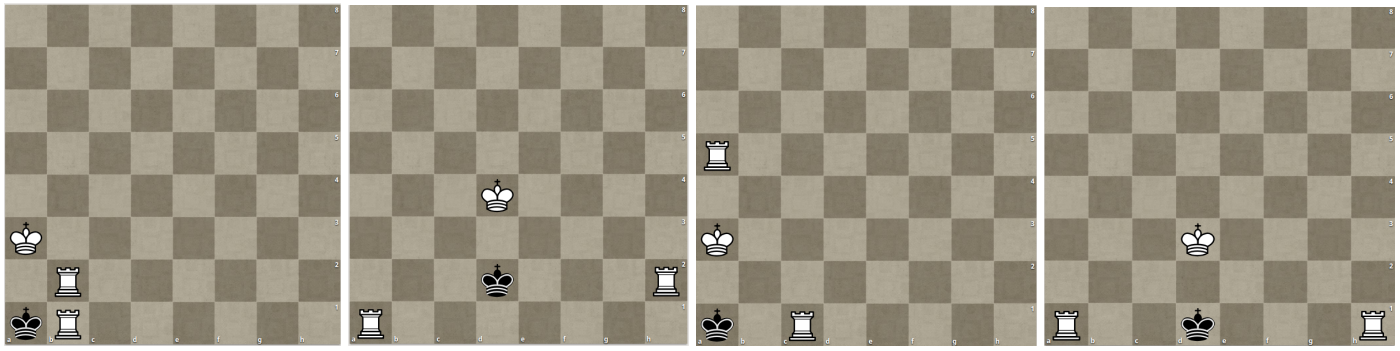
Output

For each case, output one line. First output the case number then the number of valid ways modulo **7340033**.

Sample Input	Output for Sample Input
8 1 2 3 4 5 6 7 8	Case 1: 0 Case 2: 0 Case 3: 232 Case 4: 1432 Case 5: 5188 Case 6: 14536 Case 7: 34464 Case 8: 72392

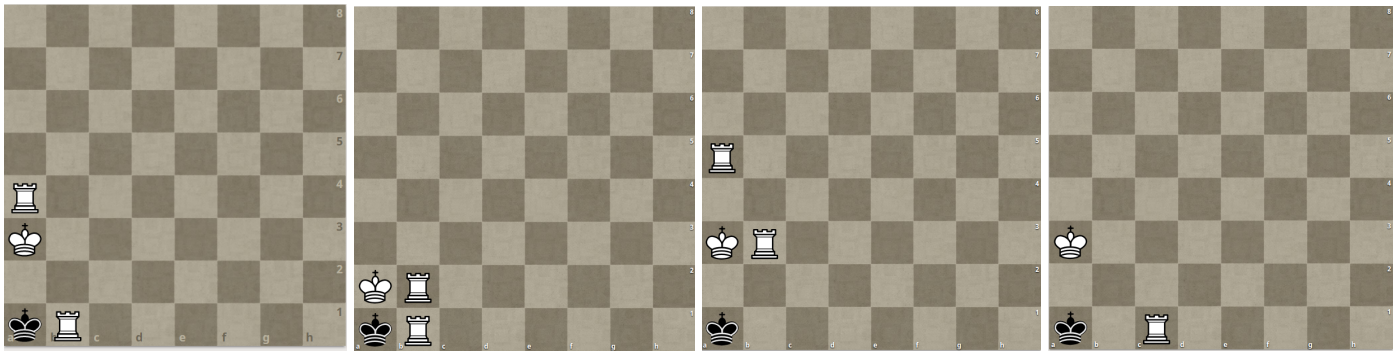
Explanation

For a **8 x 8** board, here are some valid configurations:



For all of these boards, the black king is attacked and has nowhere to go. Also both the white rooks are placed in the board and the kings are not in adjacent positions.

And here are some invalid ones:



For the first board, the black king can move and capture the attacking white rook thereby removing the check and moving to a safe square.

For the second board, although the black king is attacked and cannot capture anything because all the pieces are guarded, the two kings are adjacent which is not allowed. Hence it is not a valid checkmate position.

For the third board, it is stalemate. That is the black king has nowhere to go but it is also not attacked. So no checkmate.

For the fourth board, it is checkmate and the black king has nowhere to go. But we haven't placed both the rooks in the board and hence this is not a valid position.

Problem C

Almost A Palindrome?

You are given a string **S**, containing lowercase English letters only, and **Q** queries on it. Queries are of two types.

Query 1: (L, R)
You have to print the answer of this query in a single line. If the substring of **S** containing characters from **Lth** to **Rth** index (inclusive) is a palindrome, then the answer is 0. Otherwise you need to try to make this substring a palindrome, by removing exactly one character from it. If you can do so, then the answer for this query is the index of that character (1-based indexing) in **S**. If there are multiple such indices, then consider the smallest of them as the answer. If you cannot make the substring a palindrome by removing exactly one character, then the answer is -1.

Query 2: (X, C)
You have to set value **C** to **X-th** character in **S**.

P.S. You don't have to actually delete any characters from the string for any of the queries of type-1. This operation of removing an index is hypothetical. Queries of type-2 will persist for latter queries however.

Definitions
Palindrome: A string that reads the same backward as forward.
Substring: A contiguous sequence of characters within a string.

Input

First line of input will contain an integer **T (1 ≤ T ≤ 10)**, denoting the number of test cases. Each test case will start with a line containing a string **S**. Next line will contain an integer **Q**, denoting the number of queries to follow. Summation of **S** over all test cases, and summation of **Q** over all test cases will not exceed **200000**.

Each query will start with an integer **TYPE (1 ≤ TYPE ≤ 2)**. If **TYPE** is **1**, then it's a query of type-1, and it'll follow with two integers **L, R (1 ≤ L ≤ R ≤ |S|)**. Otherwise, it'll contain an integer **X (1 ≤ X ≤ |S|)** and, a character **C**, which is a lowercase English letter. This represents a query of type-2.

Output

For each test case, print the case number in a separate line. Then for each query of type-1, print the answer to that query according to the above definition, in a separate line. Refer to the sample I/O for more clarity on formatting.

Sample Input	Output for Sample Input
2 abca 5 1 1 4 1 2 4 2 4 w 1 1 4 1 3 3 xyz 4 1 1 2 1 1 3 2 3 x 1 1 3	Case 1: 2 -1 -1 0 Case 2: 1 -1 0

Explanation of Sample I/O

Explanation for Case-1:

Query-1: Removing character 'b' from the 2nd index will yield the substring "aca", which is a palindrome. Note that removing the 3rd index can create a palindrome as well. But we are looking for the smallest index, in case of multiple answers.

Query-2: There are no ways to make substring "bca" a palindrome.

Query-3: Now the string has become "abcw".

Query-4: Similar to query-2, there are no ways to make substring "abcw" a palindrome.

Query-5: It's already a palindrome.

Problem D

Winning Arrangements

N players are participating in a tournament. Before the tournament begins, the strength of each player is measured. In the tournament, the players are lined up in a certain order. **N-1** rounds are played. In each round any **two neighboring players** can be chosen and the stronger player wins. If both of them have the same strength then **either** can win. After the round the losing player leaves the tournament and the winning player goes back to their position. The players then fill the gap. The winner of the round also receives **1** bonus strength. A player wins the tournament if they are the last player remaining.

For each player **Pi**, find out how many initial arrangements of **N** players are there so that there is a way that player **Pi** could potentially win the tournament. Since the answer can be very large, print it modulo **1,000,000,007**.

Input

The first line of the input contains an integer T **(1 ≤ T ≤ 1,000)**, the number of test cases in the input file. Then **T** cases follow. Each test case starts with an integer **N (1 ≤ N ≤ 1,000)**, the number of players in the tournament. This is followed by a line with **N** integers corresponding to the strength of the players. Strength is a positive integer and can be at most **1,000,000,000**. It is guaranteed that ∑**N ≤ 50,000** over all cases in a single file.

Output

For each case, first print one line identifying the case number in the format **"Case X:"**. Then print **N** lines, the **i-th** line should correspond to the **i-th** player. Each line should contain an integer, the number of initial arrangements that could potentially lead to this player winning the tournament, modulo **1,000,000,007**.

Sample Input	Output for Sample Input
3 3 1 2 3 4 1 2 1 1 2 1 2	Case 1: 0 4 6 Case 2: 20 24 20 20 Case 3: 0 2

Explanation: In the second example, the first player won't be able to win in these 4 arrangements, [1, 2, 3, 4], [1, 2, 4, 3], [3, 4, 2, 1], [4, 3, 2, 1].

Note: In some slow languages (like python), it may not be possible to solve this problem.

Problem E

Alice, Bob and the Array

After a long break, we are having a programming contest in our country, and it's the preliminary contest for ICPC Dhaka Regional, 2020. Alice and Bob are here as visitors on this beautiful evening. They brought some colorful balloons for Dhaka as a token of appreciation. The host of the contest decided to keep those balloons in quarantine for next two weeks due to Covid19 issues. But, they did not want to go to the contest room empty handed and decided to present an array to the contestants.

So, Bob built an array with **N** integers within a very short time and wants to make it as beautiful as possible. He believes the beauty of an array is equal to its Maximum Consecutive Subarray Sum (MCSS) . MCSS of an array is the largest subarray sum of any contiguous non-empty subarray indexed from ***i*** to ***j*** such that $1 \leq i \leq j \leq N$.

Now, he asked Alice to perform a series of operations (if required) on the array to maximize its beauty. On each operation, she can select any two indices ***i*** and ***j*** such that $1 \leq i < j \leq N$ and reverse the subarray (***i*** and ***j*** inclusive).

As Alice is in a hurry she wants your help to do so. You have to maximize the beauty of the array performing the given operation as many times as it is required. If there are several ways to get the maximum beauty, you have to choose the one that requires the minimum number of moves.

Input

The first line of the input contains an integer **T (1≤T≤100)**, denoting the number of test cases. Each of the next T test cases has two lines of input. First line of each test case contains an integer **N (1≤N≤30000)** denoting the size of the array and the second line contains **N** space separated integers $a_1, a_2, a_3, \ldots a_N$ (**-100000**$\leq a_1, a_2, a_3, \ldots, a_N \leq$**100000**) representing the array. **Large dataset, use fast I/O methods.**

Output

For each test case print the test case number followed by the maximum consecutive subarray sum that can be achieved and the minimum number of moves. See the sample I/O section for output formatting.

Sample Input	Output for Sample Input
2 4 1 -10 2 2 2 1 2	Case 1: 5 1 Case 2: 3 0

Explanation

Case 1: Initial array is {1, -10, 2, 2}, One possible solution is as follows:
Consider i =1 and j =2 and reverse the subarray.
After performing the operation, the array will become {-10, 1, 2, 2}. So, Maximum consecutive subarray sum is 1 + 2 + 2 = 5.
Case 2: Initial array is {1, 2} and you do not need to perform any operation.

Problem F

Co Primes

Damien Rice is obsessed with songs, melodies and patterns. To create his next song, rootless tree, he has taken a keen interest in consecutive numbers. So he is trying to find a pattern in consecutive numbers but failed to do so, as he is a singer, not a mathematician or a programmer like you. But another member of his band, Lisa Hannigan, has thought about a problem involving two sequences of consecutive numbers starting from **a** and **b** respectively. Solution to this problem will help Damien to create the melody of this new song. For him, it takes a lot to solve this kind of problem. So, you have decided to help him. Now, it's your turn to fulfill one of your favorite faded fantasies of working with Damien Rice. Hurry up! Solve the following problem given by Lisa Hannigan.

Given **a**, **b** and **m**, find how many integers **i** (**0 ≤ i ≤ m)** exist such that **gcd(a + i, b + i) = 1.** Here, **gcd(x, y)** is the greatest common divisor of integers **x** and **y**.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 1000)**. Each test case will contain three integers denoting **a**, **b** and **m**. Here, **1 ≤ a, b ≤ 10^{12} and 0 ≤ m ≤ 10^{12}.**

Output

For each test case, print the case number in a single line followed by the answer. Please see the sample for details.

Sample Input	Output for Sample Input
3 1 5 2 5 12 2 10 20 1	Case 1: 2 Case 2: 2 Case 3: 1

Explanation

For the first example, two pairs exist: (1, 5) and (3, 7).
For the second example, two pairs exist: (5,12) and (6, 13).
For the third example, only one pair exists: (11, 21).

Problem G

Strange Typewriter

You are given a strange typewriter, where each key of the typewriter contains a string consisting of lowercase alphabets. Consider these strings as **keyString**. You have to type a specific string **S** using this typewriter. If it is possible to type the string using this typewriter, you have to output the minimum number of keystrokes to write the string. Otherwise, you have to output "impossible".

Input

The first line of the input gives the number of cases, **T**. Each test case starts with a line containing the number of keys in the typewriter **N**. The following **N** line includes **N keyStrings** where keyString[i] corresponds to the i^{th} key. The following line will contain a single string **S** (containing lowercase alphabets) which you have to type.

Constraints:

- **1 ≤ T ≤ 10**
- **1 ≤ N ≤ 100000**
- **1 ≤** Summation of all |keyString| **≤ 1000000**
- **1 ≤ Summation of all |S| ≤ 1000000**
- **1 ≤ |Any string in input| ≤ 100000**

Output

For each test case, if it is possible to write, then print the minimum number of keystrokes required to write the string **S;** otherwise, print "impossible". See the sample outputs for the exact format.

Sample Input	Output for Sample Input
2 3 a b aab aab 2 a ba aab	Case 1: 1 Case 2: impossible

Problem H

Recursion, Lambda and Parameter

Mrs. Recursion runs a brokerage company. One day she came up with an idea of the custom fund offer. There are **N** types of stocks available to exchange in her brokerage. Each stock has two attributes. For the i'th stock they are: its price $\mathbf{P_i}$ and the average return $\mathbf{R_i}$. Now sometimes some of the stocks have such a high price that an individual may not afford it. So Mrs. Recursion came up with this revolutionary idea of the custom fund. Instead of buying a single stock, an individual may mix up the stocks. The price of this mixed up stock will have the weighted price. The average return of it will be the weighted average return of the constituent stocks. We call this mixed up stock- a fund. For example, someone may form one fund taking 0.5 stock from the first one, 0.3 from the second one and 0.2 from the third one (please note these fractions need to sum up to 1). Then the price of this fund is: $0.5P_1 + 0.3P_2 + 0.2P_3$ and the average return of this fund is $0.5R_1 + 0.3R_2 + 0.2R_3$. Needless to say, one can form a fund with one stock only, if they wish. For example one can form a fund just from the first stock and its price would be P_1 and the average return would be R_1.

Now here comes the Lambda with a brilliant consultancy idea. This is his advertisement:

You want to spend at most P money per fund?
I can tell you how to get the maximum average return.
You want to get at least R average return?
I can tell you what's the minimum fund price to achieve that.

However, Lambda does not know how to solve these problems. So he wants to hire a bunch of Parameters among you in this International Collegiate Parameters Competition.

Input

First line contains the positive integer **T**, the number of test cases. The first line of each test case starts with two positive integers **N**, the number of stocks and **Q**, the number of queries. There are **2N** non negative integers in the following line in the format of: $\mathbf{P_1\ R_1\ P_2\ R_2 \ldots P_N\ R_N}$**.** **Q** lines follow. Each line will be of one of the following formats:

1 P M: For at most **P** money per fund, what's the maximum average return one can achieve if one is allowed to mix at most **M** stocks to form the fund
2 R M: To achieve at least **R** average return, what would be the minimum fund price if one is allowed to mix at most **M** stocks to form the fund.

Constraints:

$1 \le T \le 10$
$1 \le N, Q \le 100{,}000$
$0 \le P_i, R_i, P, R \le 1{,}000{,}000{,}000$
$1 \le M \le N$.

Output

For each test case print the case number. Followed by the answers to the queries in different lines in the reduced fraction form X/Y (there is no common positive divisor of X and Y other than 1). Since X and/or Y may be too big, print them modulo 2^{64}, that is, you will actually print (X mod 2^{64})/(Y mod 2^{64}). Please note, if the answer is an integer, Y should be 1. It may be the case that the solution does not exist. In such a case print 0/0. For clarity check the sample input-output.

Sample Input	Output for Sample Input
3 2 3 5 10 5 20 1 4 2 2 5 2 2 15 2 2 3 5 5 10 5 1 6 1 1 6 2 2 10 1 2 1 10 10 20 20 1 15 2	Case 1: 0/0 5/1 5/1 Case 2: 5/1 5/1 0/0 Case 3: 15/1

Explanation

In the third case, there are two stocks. If we weight the stocks as 0.5, our fund price would be: 0.5 * 10 + 0.5 * 20 = 15 which yields 0.5 * 10 + 0.5 * 20 = 15 average return. It turns out this is the maximum average return with a fund price of 15.

Problem I

Lattice Triangle

In the Cartesian coordinate system a lattice point is a point whose abscissa and ordinate are both integers. For example (3, 4) is a lattice point, but (3.5, 4) or (√2, √3) are not lattice points. A lattice polygon is a polygon whose vertices are all lattice points. There is an unknown lattice triangle **TRI**. It has **L** lattice points inside (Points on the boundary are not considered inside). If the coordinates of all its vertices (both abscissa and ordinate) are multiplied by **n** (a rational number given in the form **p/q**, here **p** and **q** are positive integers) we get another lattice triangle (the value of n will be such that we will get another lattice polygon). The number of lattice points inside this new lattice triangle is $\mathbf{L_2}$. Given the value of **L,** $\mathbf{L_2}$ and n you have to find the area of **TRI**. You can assume that for the given values there will always be two such lattice triangles.

Input

There will be at most **20000** cases. Each line contains two positive integers which denotes the value of **L (0 ≤ L ≤ 100000)** and $\mathbf{L_2}$ **(0 ≤** $\mathbf{L_2}$ **≤** $\mathbf{10^{13}}$**)** and a fraction in the form **p/q** (Here **p** and **q** are positive integers and **p ≤ 10000** and **q ≤ 500** and **p** and **q** are not equal). Input is terminated by a line where all integer values are zero. This line should not be processed.

Output

For each line of input produce one line of output. This line should contain an integer that denotes twice the area of **TRI**.

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

Problem J

COVID - 19

COVID - 19 has been the most prominent topic in our lives for the last year. Just when some of us thought that the pandemic was under our control, we are going through another huge increase in infection rate. There is still very little data to predict what is going to happen with this virus. So as general people we should really listen to the experts. We should wear a mask whenever we are interacting with another person. We should avoid unnecessary movements completely. We should sanitize our hands frequently. Most importantly, we should keep our distances. Recommended is six feet. This distance keeping among humans is being called Social Distancing.

Let's say there is a corridor with length N feet. The width of the corridor is really narrow. For simplification we can imagine the corridor as a line of length N. Let's also assume that each human can be imagined as a point in this corridor. What is the maximum number of humans that can be in the corridor maintaining Social Distancing? For example if N = 6, then we can have two humans on two opposite ends of the corridor. If N = 12, then we can put three humans safely, two on the end points, and one in the middle of the corridor. But if N = 5, then we have no way to put more than one human in the corridor. Because we won't be able to maintain six feet distance between them.

Input

Input will have a single integer **N ($1 \leq N \leq 10^4$)**, denoting the size of the corridor in feet.

Output

Print the maximum number of people you can put in the corridor maintaining Social Distancing.

Sample Input 1	Output for Sample Input 1
5	1

Sample Input 2	Output for Sample Input 2
6	2

Sample Input 3	Output for Sample Input 3
12	3

Sample Input 4	Output for Sample Input 4
13	3

icpc international collegiate programming contest

ICPC Asia West Regional 2019

ICPC Asia Dhaka

Regional Contest

Official Problem Set

icpc.foundation

icpc global programming tools sponsor

SSL WIRELESS©

16th November 2019

You get 15 Pages, 10 Problems & 300 Minutes

Problem A

Lord of the Curses

Talion, blessed with the power of the Bright Lord is preparing an attack against the Dark Lord. But the Dark Lord has an army of **N** Uruks. Before having a face to face war with the Dark Lord, Talion has to defeat a few Uruks. Let's consider the army of Uruks are lined up from left to right.

1st Uruk 2nd Uruk 3rd Uruk 4th Uruk 5th Uruk ..... Nth Uruk

Each Uruk has a maximum health capacity $\mathbf{H_i}$ and present health $\mathbf{PH_i}$ ($\mathbf{1 \le i \le N}$). Before any attack, Talion also gets a new health $\mathbf{PH_T}$. Some Uruks serve other Uruks as **bodyguard**. When the **i'th** Uruk serve the **j'th** Uruk as a bodyguard, we call the **j'th** Uruk the boss of the **i'th** Uruk. An Uruk can be the boss of multiple Uruks, but an Uruk can serve at most one Uruk as bodyguard at a time.

At any moment during the attack, if the **i'th** Uruk is **alive** ($\mathbf{PH_i > 0}$) and is **not cursed** by the dark lord, the boss of the **i'th** Uruk is found using the following rule:

The **j'th** Uruk will be the boss of the **i'th** Uruk if $\mathbf{i < j}$, $\mathbf{H_i \le H_j}$, $\mathbf{(j - i)}$ is minimum and the **j'th** Uruk is alive. If there's no such Uruk who can be considered as the boss of the **i'th** Uruk, then the **i'th** Uruk directly serves the Dark Lord.

During the attack, the Dark Lord might follow some tactics to improve his army. Sometimes he might curse few of his Uruks and sometimes he might bless a few of them as well.

Sometimes the Dark Lord may bless an Uruk. If the Uruk was dead or cursed before, it becomes alive now and also gets rid of all the previous curses due to the blessing. Besides, it also gets a new **maximum health capacity** and it'll revive its present health to this new maximum health capacity as well. Additionally, it will now find his new boss using the above rule. At the same time, some other Uruks (who are currently **alive** and **not cursed** by the Dark Lord) might choose the blessed Uruk as their new boss now using the above rule.

Sometimes, the Dark Lord may become angry with some Uruks and then he curses them. If the Dark Lord curses the **i'th** Uruk, he will force the **i'th** Uruk to serve the **j'th** Uruk (where $\mathbf{i < j}$ and the **i'th** or **j'th** Uruk might be dead as well during that time) as **bodyguard**, even if $\mathbf{H_i > H_j}$. Due to this curse, the **i'th** Uruk will continue serving the **j'th** Uruk until the **i'th** Uruk is blessed or cursed again by the Dark Lord. But the curse doesn't affect any changes on the health of the **i'th** or **j'th** Uruk and also, it doesn't make any changes on the dead or alive state of any of the them. A cursed Uruk can be the Boss of other Uruks as well.

Talion starts an attack with the health $\mathbf{PH_T}$. Usually an attack consists of multiple fights between Talion and Uruks but he fights against one Uruk at a time. Initially, he selects an Uruk (who might be already dead) to start a fight and then continue fighting with some other Uruks in the process until he's dead or reaches to the Dark Lord.

When Talion is in a fight with the **i'th** Uruk, Talion will have **max(0, PH_T - PH_i)** health left after the fight and the **i'th** Uruk will have **max(0, PH_i - PH_T)** health left. If at any time, the present health of the **i'th** Uruk or Talion becomes **0,** they die instantly. If Talion survives the fight, he will go forward to face the boss of the **i'th** Uruk (even if it is already dead), and so on. But if he dies, the attack ends here. Also, if the **i'th** Uruk is dead after this fight, then all of its bodyguards (who are still alive and not cursed) will find their new boss according to the rule described above.

Talion always do stealth attack. As a result, none of the bodyguards of an Uruk gets notified about the fight. He reaches to the Dark Lord **only when** he defeats an Uruk who serves directly to the Dark Lord.

As Talion is blessed by the Bright Lord, he reborns again after each death and also revives his health to a new capacity and prepare for the next attack.

Initially all the Uruks are alive with their full health (**PH_i = H_i)** and not cursed. There will be four types of queries depending on what's going on in the Middle earth:

1. Talion initiates an attack with the health **PH_T** and select **X'th** Uruk to start the fight. He continues the attack following the process explained above until he's dead or faces the Dark Lord. Print the amount of health Talion will have when he faces the Dark Lord. If Talion dies before reaching the Dark Lord, print **0**.
2. The **X**-th Uruk is cursed because of the anger of the Dark Lord and is forced to serve as the bodyguard of the **Y**-th Uruk (**X < Y**).
3. The **X**-th Uruk is blessed by the Dark Lord and gives him a new health capacity **Z**.
4. Print how much health **X'th** Uruk has right now i.e. the value of **PH_X**.

Input

The first line of the input contains an integer **T** which denotes the number of test cases.
The first line of each test case contains two integers **N** and **Q** where **N** is the number of Uruk in the Dark Lord's army and **Q** is the number of queries you need to process. The second line contains **N** integers **H_1 H_2 H_3 … H_N** each denoting the maximum health capacity of the Uruks. Each of the next **Q** lines contains any one type of queries which are described above in the following format:

1 X PH_T: First type of query **(1 ≤ X ≤ N)**
2 X Y: Second type of query **(1 ≤ X < Y ≤ N)**
3 X Z: Third type of query **(1 ≤ X ≤ N)**
4 X: Fourth type of query **(1 ≤ X ≤ N)**

Constraints

- **1 ≤ T ≤ 5**
- **1 ≤ N, Q ≤ 100,000**
- **1 ≤ Z, PH_T, H_i ≤ 1,000,000,000**

Output

For each test case, the first line should contain the case number in the format "**Case X:**" (without the quotes) where **X** is the case number. After that for query type of **1** and **4**, you need to print the answers.

Sample Input	Output for Sample Input
2 5 5 5 2 1 3 4 1 3 9 2 2 4 4 4 1 2 4 4 1 3 4 1 2 3 1 2 2 1 1 2 3 2 5 1 1 3	Case 1: 1 0 2 5 Case 2: 0 0 1

Problem B

I know Recursions!

Toru and her friend Lota were preparing themselves for the ICPC Asia Dhaka Regional Contest. They learned about recursive functions and started practicing. One of the first things that people do while learning recursions is to generate the *Fibonacci series*. In case you are not familiar with this, in the *Fibonacci series*, each number except the first two is defined as the sum of the two numbers which came just before it. The first few numbers of this series are 0, 1, 1, 2, 3, 5, 8, 13, 21 … and so on. In terms of functions, the ***N-th*** Fibonacci number can be written as:

```
Fibonacci(0) = 0 and Fibonacci(1) = 1
Fibonacci(N) = Fibonacci(N-1) + Fibonacci(N-2), for N > 1
```

They too practiced the implementation of this series. However, their curious minds did not stop just there. Together, they came up with another series where each number would be the sum of ***P*** preceding numbers instead of just **2**. They named it as the "*P-bonacci series*"! Then they wrote the pseudocode of a recursive function to generate the ***N-th*** *P-bonacci number*.

```
P_bonacci(N, P)
    if N < P
        return N

    ans := 0
    for each i between 1 to P
        ans += P_bonacci(N-i, P)

    return ans
```

Being excited about their discovery, Toru and Lota went to talk to their teacher about this. The teacher was proud of her students and wanted them to learn more. So she came up with a task for them. To compute the *N-th P-bonacci number*, the function above will perform some recursive calls. She told them to figure out the *caller* and *callee* of the ***K-th*** call (considering only the parameter ***N***). For example, if we start with ***N = 5*** and ***P = 3***, the first recursive call will be from ***N = 5*** to ***N = 4***. And the next few calls in order would be 4 → 3, 3 → 2, 3 → 1, 3 → 0, 4 → 2, 4 → 1, 5 → 3 and so on. So in the first recursive call, *caller* is **5** and *callee* is **4**. Similarly in the 4th recursive call, *caller* is 3 and *callee* is 1.

Toru and Lota are smart and enthusiastic enough to solve this problem eventually. But can you solve this today?

Input

First line of input will contain **T (1 ≤ T ≤ 1000)** denoting the number of test cases.
Each of the next T lines will contain three space separated integers **N (1 ≤ N ≤ 10000), P (1 ≤ P ≤ 1000) and K (1 ≤ K ≤ 10^{17})** as described above.

Output

Print a single line for each case. First, print the case number **X** in the format "**Case X:** ". Then print two integers **A** and **B** which are the *caller* and the *callee* of the ***K-th*** recursive call, respectively. If, **K** calls will not be required for the given values of **N** and **P**, then print "**Invalid**" instead of **A** and **B**. See the sample below for more clarity.

Sample Input	Output for Sample Input
3 4 2 5 8 2 600000 80 30 1000000000000	Case 1: 3 1 Case 2: Invalid Case 3: 30 10

Explanation of Case 1:
Recursive calls in order: 4 → 3, 3 → 2, 2 → 1, 2 → 0, **3 → 1**, 4 → 2, 2 → 1, 2 → 0.

Problem C

Colored Development

There are **N** unique colors in the universe, numbered from **1** to **N**. George Michael wants to create a rainbow using these colors. The rainbow will consist of exactly **M** layers. For each layer, George Miachel selects a color uniformly randomly between **1** to **N** and then paints the layer with it.

George Michael wonders what will be the **expected** number of **distinct** colors in the rainbow after all the layers are colored in this way.

Input

The first line of the input contains an integer **T**, denoting the number of test cases. The next **T** lines will contain two integers, **N** and **M**.

Constraints

- **$1 \le T \le 10$**
- **$1 \le N, M \le 2 \times 10^5$**

Output

For each test case, print the case number and the expected number of distinct colors in the rainbow after all the layers are colored. Formally, let the expected number of distinct colors be an irreducible fraction **P / Q**. Then you need to print **$P \times Q^{-1}$ modulo 1000000007**, where **Q^{-1}** is the modular inverse of **Q** modulo **1,000,000,007**.

Sample Input	Output for Sample Input
3 1 1 2 2 4 2	Case 1: 1 Case 2: 500000005 Case 3: 750000007

Explanation

For the second test case where **N = 2** and **M = 2**,

Let **Pr(X)** be the probability to get **X** distinct colors after all the layers are colored.
Expected number of distinct colors = **$Pr(1) \times 1 + Pr(2) \times 2 = ½ \times 1 + ½ \times 2 = 3/2$**

Here, **P = 3** and **Q = 2**
Q^{-1} modulo 1000000007 = 500000004
$P \times Q^{-1}$ modulo 1000000007 = $3 \times 500000004 \% 1000000007 = 500000005$

Problem D

Array Permutation

"The ICPC is just like the Olympics of programming competitions, the oldest, largest, and most prestigious programming contest in the world."

Alice and Bob are two of the most important characters that you must have already heard if you are a contestant. Alice and Bob are good friends. They always keep fighting; keep each other busy with different puzzles or strategy games. None of them want to lose. So whenever they can't find a solution they come to a programming contest and ask the contestants for help.

Alice and Bob were playing a new game called the game of permutation. The game starts with an array consisting of **N** elements from **1** to **N**. At the start of the game Alice generates a permutation of the array. She gives Bob **Q** constraints about the array in the following format.

L R X: the minimum value of the array within the range **L** to **R** inclusive is **X**.

Finally, Bob has to answer whether such an array is possible or not. After a few games, Alice identified that it's too easy for Bob. So, she decided to make things harder for Alice. She asked Bob "How many permutations of the array will satisfy the **Q** constraints?".

Bob is not good at mathematics. Since It's the ICPC Dhaka Regional, 2019 and Bob knows you are a good programmer, he is asking for your help. Bob promises to give you one balloon if you can solve it for him.

Input

Input starts with an integer **T (1 ≤ T ≤ 10**) denoting the number of test cases.

The first line of a test case will contain two integers **N (1 ≤ N ≤ 50000)** and **Q (0 ≤ Q ≤ 10^6)** in a single line. Next **Q** lines contain three integers **L, R(1 ≤ L ≤ R ≤ N)** and **X(1 ≤ X ≤ N)**.

Summation of **Q** over all test cases **≤ 10^6**.

Output

For each test case, print the case number followed by the desired answer. Since the answer can be very large, print the answer modulo **1,000,000,007 (10^9 + 7).**

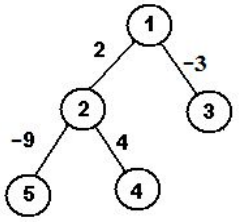
Sample Input	Output for Sample Input
2 5 2 3 4 1 4 4 1 3 1 1 3 2	Case 1: 24 Case 2: 0

Problem E

Trees Are Beautiful

A tree with **N** number of vertices has **N-1** edges connecting them. Each edge has an integer value associated with it known as weight. The distance between any two vertices **u** and **v**, denoted by **D(u, v)** is the summation of the weights of the edges in the path between them.

For more clarity, consider the given tree in the figure. There are five vertices numbered from **1** to **5** and there are four edges. Vertices **(1, 2), (1, 3), (2, 4), (2, 5)** are connected by edges having weight **2, -3, 4** and **-9** respectively. So, the distance between vertices **1** and **2**, **D(1, 2) = 2**, distance between vertices **1** and **5**, **D(1, 5) = -7** and the distance between vertices **5** and **3**, **D(5, 3) = -10**.



A tree is said to be ***Beautiful*** if the summation of all pair distance of the vertices of the tree is non-negative. The summation of all pair distance for the given tree = ***D(1 ,2) + D(1, 3) + D(1, 4) + D(1, 5) + D(2, 3) + D(2, 4) + D(2, 5) + D(3, 4) + D(3, 5) + D(4, 5) = -20.*** Note that, for any two vertices **u** and **v** where **u** ≠ **v,** the distance is considered once.

You are given a tree. You have to determine whether the given tree is Beautiful or not. If the tree is not Beautiful, you have to perform a series of operations to make given trees beautiful. The operation is as follows: **Select an edge whose weight is negative and increase its weight by 1.**

Now you have to determine the minimum number of times you need to perform such an operation to make the given tree Beautiful. Can you answer it?

Input

Input begins with an integer **T(1 ≤ T ≤ 30)** denoting the number of test cases.

The first line of a test case will be an integer **N(2 ≤ N ≤ 20,000)** denoting the number of vertices in the tree. Next **N-1** lines will contain three integers **u(1 ≤ u ≤ N)**, **v(1 ≤ v ≤ N, u ≠ v)**, and **w (|w| ≤ 10^5)** denoting there is an edge connecting vertices **u** and **v** and the weight of the edge is **w**.

Output

For each test case, print the case number followed by the desired answer.

Sample Input	Output for Sample Input
2 5 1 2 2 1 3 -3 2 5 -9 2 4 4 2 1 2 1	Case 1: 5 Case 2: 0

Explanation:

Case 1: A solution can be, increase the weight of the edge connecting vertices (2, 5) five times.

Problem F

What happens if you sum?

Find the sum of the Lowest Common Multiples (LCM) of each pair of integers from a given **N** positive integers.

Input

The first line of the input contains an integer **T**, denoting the number of test cases. Each test case consists of two lines. First line contains **N**. Second line contains **N** positive integers separated by a single space.

Constraints

- **1 ≤ T ≤ 10**
- **1 ≤ Any number in the input ≤ 100,000**

Output

For each test case, print the case number followed by the required sum. Since the resulting sum can be very large, output the sum modulo **132021913**.

Sample Input	Output for Sample Input
2 2 1 10 3 2 3 4	Case 1: 10 Case 2: 22

Problem G

Where are Nordomas Kit?

Recently we found a new type of bacteria Nordomas Kit attacking Hbs, one of the most beautiful trees in Teub, our garden. These bacteria attack just the trunk of the tree. The long trunk of the tree can be represented by a long grid of **2** rows and **n** columns. Each cell of the grid contains a nutrition value. For example, for **n = 5**, the grid may look like this:

5(a)	3	8	2	0(c)
1(a)	12(b)	1(b)	9	7(c)

Nordomas Kit are **2** cells long. A Nordomas Kit occupies two adjacent cells, either horizontally or vertically. It may not lie outside of the grid, neither fully nor partially. Two Nordomas Kit may not occupy the same cell but they may occupy two neighboring cells. For example in the above grid, we placed three Nordomas Kit a, b and c. a and c occupy two vertical cells while b occupies two horizontal cells. a and b are touching.

We know there are **k** Nordomas Kit in this whole grid. They occupy the cells in such a way that the sum of the nutrition value in the occupying cells is maximum. Find this maximum sum of the nutrition value.

Input

The first line of the input contains an integer **t**, denoting the number of test cases. Each test case consists of four lines. First line contains two integers: **n** and **q**, the size of the grid and the number of queries. Following two lines contains **n** non negative integers each, which represents the grid. The next line contains **q** positive integers, denoting the values of **k**.

Constraints

- **1 ≤ t ≤ 5**
- **1 ≤ n ≤ 100,000**
- **1 ≤ q ≤ 10**
- **The nutrition values in the grid cells are not more than 1,000,000.**
- **1 ≤ k ≤ n**

Output

For each test case, print the case number followed by **q** integers, answers for each of the query.

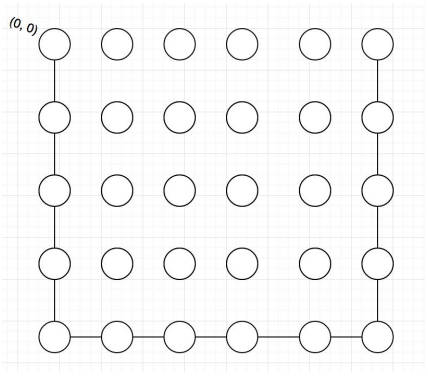
Sample Input	Output for Sample Input
1 5 5 5 3 8 2 0 1 12 1 9 7 5 1 3 2 4	Case 1: 48 16 41 31 47

Problem H

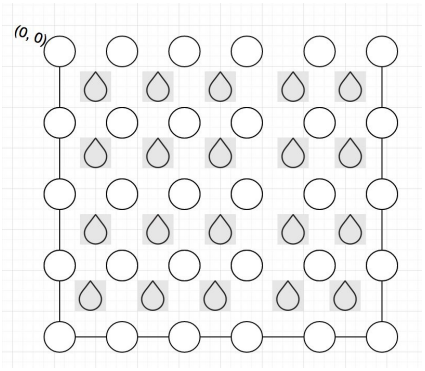
Droplets

Climate change is real, it will affect all of us sooner or later. This problem is about saving water and you will be a climate hero if you can solve it.

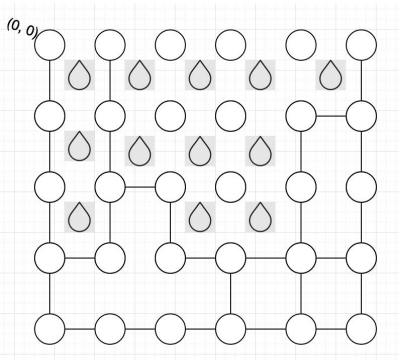
We have a matrix which consists of **N** rows and **M** columns.. Consider every cell of the matrix as a dot. The top left cell of the matrix is (0, 0). Initially it looks something like Picture 1.



Picture 1



Picture 2



Picture 3

In picture 1, all the sides of the matrix are enclosed by straight lines except the top. That is because we want to collect water inside the matrix!

Imagine it's continuously raining outside and the water falling from the top of the matrix. Every square sized area surrounded by four dots can contain precisely one unit of water. The distance between every cell is equal. If we pour water in the matrix of Picture 1, it will look something like Picture 2. This matrix contains **20** units of water.

Now some of the cells of the matrix can also be connected by straight lines too, as you can see in Picture 3. As the water can't go through these lines, this matrix is only able to contain **12** units of water.

Now your task is simple. Given the matrix, find the amount of water it can hold. You can assume that the left, bottom, and right sides of the matrix will always be enclosed. Imagine the matrix is in a vacuum, that is once water is inside it will flow in all directions without considering factors like air pressure.

Input

The first line contains an integer **T (1 ≤ T ≤ 100)**, denoting the number of test cases.
The first line of each case contains two integers **N** and **M (2 ≤ N, M ≤ 100)**, denoting the size of the matrix.
Next there are **N** lines, each containing **M** characters denoting the matrix. Each cell of the matrix will contain exactly one of the four characters **'N', 'R', 'D', 'B'**.

'R' means the cell is connected to the cell on the right, **'D'** means the cell connected to the cell underneath, **'B'** means the cell is connected to both it's right and underneath cells. **'N'** means there is no connection.

Output

For each case, print the case number and the amount of water the matrix contain.

Sample Input	Output for Sample Input
3 4 5 DDDND DBNND BDBRD RRRRN 5 6 DDNNND DDNNBD DBDNDD BNRBBD RRRRRN 4 4 BBDD DDDD DRND RRRN	Case 1: 9 Case 2: 12 Case 3: 7

Problem I

Round Tables

You have a table of convex polygon shape. The table has **N** corners and there are seats at each of these points. You can imagine the position of these seats as a point in the **2D** euclidean coordinate system. You need to put two bone-plates (plates to put the bones, seeds, etc.) in the table. The position of these bone-plates should be inside the table in such a way so that the maximum distance between a seat and any of the two bone plates is minimized. You can assume the plates as **2D** points as well. You need to find the optimal positions of the plates.

Input

First line contains an integer **T** (**1 ≤ T ≤ 10**) denoting the test cases. Each case will start with an integer **N (4 ≤ N ≤ 500)**, the number of seats. Each of the next **N** lines, will contain two integers **Xi, Yi (0 ≤ |Xi|, |Yi| ≤ 1000)** which indicates the position of the i-th seat. The seats will be given in clockwise/anti-clockwise order.

Output

For each case, you need to print the case number then output the position of the two bone-plates. While outputting positions, first print the **X** and then the **Y** of each bone-plate. If there are multiple solutions, print any such solution. Absolute errors less than 10^{-4} will be ignored. While calculating error, the maximum distance between any seat and your solution's bone-plate positions will be compared with the optimal maximum distance.

Sample Input	Output for Sample Input
1 4 0 0 10 0 10 5 0 5	Case 1: 0 2.5 10 2.5

Note

Input for this problem was generated randomly following the process below:

1. Huge number (at most **250,000**) of random points were generated using the following strategy:
   a. **X** was chosen randomly in the range **[-1000, 1000]**.
   b. **Y** was chosen randomly in the range **[-1000, 1000]**.
2. Convex hull of those points were calculated (keeping all the points that lie on the border).
3. If the number of points on the convex hull is between **4** to **500**, then this is included as an input case.

SSL WIRELESS

Problem J

Greatest GCD Ever

Bangladesh Association of Problem Setters

GCD means greatest common divisor. Given two integers **A** and **B**, **GCD(A, B)** is the **LARGEST** integer that divides both **A** and **B**. For example, **GCD(10, 15)** is **5**, **GCD(21, 35)** is **7** etc.

Given **A** and **B**, it's not difficult to find **GCD(A, B)**. But what if you didn't know both of them? Lets say, you know the value of **A**, but for **B**, you know the possible range of values **[0, N]** which can be given as value of **B**. That means value of **B** can be any integer starting from **0** to **N** (inclusive). Can you find the maximum **GCD(A,B)** for the given **A** considering all possible **B**?

For example, if **A = 12** and **B** has a possible range **[0, 100]**, then the maximum GCD that can be made is **12**. One of the ways we can do it is by assigning **24** as the value of **B**.

Input

Input will have two integers **A (-100 ≤ A ≤ 100, A ≠ 0)** and **N (1 ≤ N ≤ 100)** in a line.

Output

Output a single integer which indicates the value of maximum GCD that can be obtained.

Sample Input	Output for Sample Input
12 100	12
15 30	15
20 10	20

SSL WIRELESS

Problem A

Almost Pattern Matching II

Bangladesh Association of Problem Setters

Two strings are defined to be *almost equal* if they are of the same length and their hamming distance is less than or equal to **K.** What this means is that they can have at most **K** mismatches.

Given a text string **S**, a pattern string **P**, and an integer **K**, count how many substrings of **S** are *almost equal* to the pattern **P** with hamming distance at most **K.**

Input

The first line of the input contains an integer **T**, denoting the number of test cases. The next **T** lines will contain the test cases. For each test case, **S**, **P** and **K** will be given in a single line, separated by a single space and in that order.

Constraints

- **1 ≤ T ≤ 10**
- **1 ≤ |S|, |P| ≤ 1200000**
- **0 ≤ K ≤ 4**

S and **P** will consist of lowercase english letters **(a-z)** only. The sum of all the characters (number of characters in text string and the number of characters in the pattern string across all the test cases) in a single input file will not exceed **2500000.**

Output

For each test case, print the case number followed by the count of substrings in **S** which are almost equal to the pattern **P** with hamming distance at most **K**. Refer to the sample input/output for more clarity on the format.

Sample Input	Output for Sample Input
5 abbabaabbaababbabaababbaa abbab 0 abbabaabbaababbabaababbaa abbab 1 abbabaabbaababbabaababbaa abbab 2 abbabaabbaababbabaababbaa abbab 3 abbabaabbaababbabaababbaa abbab 4	Case 1: 2 Case 2: 8 Case 3: 11 Case 4: 14 Case 5: 19

SSL WIRELESS

Problem B

The Social Network

Bangladesh Association of Problem Setters

Alice wants to build a new social networking site. And you are hired to implement the functionalities of the site.

There are **N** users in the site. Every user can upload posts in the site and he/she can see his/her posts by default. But if a user wants to see someone else's post then both of the user must be in the same network. Two users will be in the same network if they are friends or friends' friends or friends' friends' friends and so on. Once two user **U** and **V** are in the same network then user **U** can see all the posts that user **V** had previously uploaded and going to upload later and vice versa.

You are given **Q** queries. You need to perform the queries in the given order. Each query can be one of these types.

- **0 U V**, means user **U** and user **V** are now friends.
- **1 U T**, means user **U** uploads a post on the site in time **T**. Different users can upload posts at the same time. Also a user can upload multiple posts at the same time.
- **2 U L R**, Count the number of posts user **U** will see from time **L** to **R** inclusive. User **U** will see a post if that post is uploaded between time **L** to **R** inclusive and is uploaded by user **U** or some other user who is in the same network with user **U**.

Input

The first line contains an integer **TC (1 ≤ TC ≤ 4)** denoting the number of test cases. Each test case starts with two space separated integers **N (1 ≤ N ≤ 10^5), Q (1 ≤ Q ≤ 3*10^5)** denoting the number of users and number of queries respectively. Next **Q** lines contains one of these types of queries.

- **0 U V (1 ≤ U, V ≤ N, U ≠ V)**
- **1 U T (1 ≤ U ≤ N), (1 ≤ T ≤ 10^9)**
- **2 U L R (1 ≤ U ≤ N), (1 ≤ L ≤ R ≤ 10^9)**

Output

For each test case, print the case number and for each query of type **2 U L R**, print the number of posts that user **U** will see from time **L** to **R** inclusive in a new line. You can safely assume that the result of a **2 U L R** type query will not be affected by any subsequent queries. Please see sample for details.

Sample Input

```
2
3 6
1 2 7
2 1 1 10
0 2 1
2 1 1 10
1 2 1
2 1 1 10
3 7
1 3 2
1 1 100
1 1 2
0 1 3
0 2 1
2 2 1 10
2 1 2 2
```

Output for Sample Input

```
Case 1:
0
1
2
Case 2:
2
2
```

SSL WIRELESS

Problem C

ICGeSi Standings

Bangladesh Association of Problem Setters

"Programming contest fosters creativity, teamwork, and innovation in solving critical problems, and enables students to test their ability to perform under pressure."

ICGeSi maintains a simplified contest ranklist that has only 3 columns consisting of team id, the number of problems solved by that team and their total penalty time. To make the contest more thrilling, the contest ranklist is frozen for the last 1 hour, no update is shown.

The contest, ranklist and result is determined by the following rules:

- The team with the higher number of solved problems will get a higher rank. Rank 1 is considered as the highest rank.
- In case of a tie between two teams with the same number of problems solved, the higher rank is determined by the penalty time. The team with the lower penalty time will get the higher rank.
- The penalty time for a particular problem is determined by the following formula:
  - If a problem is not solved, then the penalty time for that problem is **0 (zero)**.
  - The time consumed for a solved problem to be solved is considered as its penalty time. For example, if a problem is solved at the 20^{th} minute of the contest, then the penalty time for that problem is, P = 20.
  - Total Penalty for a team is the summation of penalties for all problems.
    $P_{TEAM} = P_A + P_B + P_C + \ldots$, where problems are named as A, B, C, … in alphabetic order.
  - A team can not submit for a particular problem after getting it accepted once.
- After considering the number of solved problems and penalty time, if there is still a tie, teams will share a common rank. For example, let's assume that, both TeamX and TeamY solved 2 problems with total penalty time $P_{TeamX} = P_{TeamY} = 220$. It is a tie. So, they will share a common rank. In the ranklist, TeamX may come before TeamY, or TeamY may come before TeamX. Both are valid. The same also holds when more than 2 teams are tied.
- You may safely assume that each team solved at least one problem before the ranklist was frozen.

It is ICGeSi DRPC-2019 (Dhaka Regional Premier Contest, 2019). This is one of the biggest competitions in the history of DRPC and is a high voltage contest. All the problems are critical and require a lot of analysis and coding skills to be solved. For any team anything is possible, the *agony* and the *ecstasy*.

The contest has already finished and the jury is ready to unfreeze the rank list and publish the result. The jury of the contest posted an interesting statistic: "**in ICGeSi DRPC-2019, there is no such team who got an accepted verdict after any of their submissions had been rejected during the frozen hour.** " For example, let's assume there are 5 submissions from a particular team in the frozen hour. If their 2nd submission was rejected, we know for sure that the following 3 submissions were rejected too.

Everyone is eagerly waiting for the result. The organizer decided to arrange a closing ceremony in order to address the winners, participants, supporters, owners, sponsors and stakeholders. It's been five hours after the contest, but the ceremony is yet to start. A lot of rumours are going on. A lot of criticizations are also being made. From the judges' perspective, there are two types of critics: positive critics, negative critics. Negative critics are claiming that the jury is taking time to actually manipulate the results, while positive

critics are claiming that the organizers are not ready as the guests are yet to arrive. A group of people is also claiming that they have the result which has been leaked from judgeroom.

You are given the frozen ranklist of ICGeSi DRPC-2019. For each team you also know when they submitted solutions during frozen hour but you do not know their verdicts (whether they were accepted or rejected). You are also given the result which is claimed as being leaked from the judges room. You have to determine whether such a final ranklist is possible or not!

Input

Input starts with an integer **K** (**$1 \leq K \leq 750$**) in a single line denoting the number of test cases. Each of the **K** test cases starts with an integer **N** (**$1 \leq N \leq 50$**) denoting the number of teams participating in the contest. Each of the next **N** lines describe teams starting with four integers **Id (**$1 \leq Id \leq N$**)**, **S (**$1 \leq S \leq 120$**)**, **TP (**$1 \leq TP \leq 7260$**)** and **M** (**$0 \leq M \leq 60$**) denoting team id, number of problems solved by them, their total penalty time before the ranklist was frozen and the number of submissions during the frozen hour respectively. Then in the same line, there will be **M** integers T_1, T_2, …, T_M (**$121 \leq T_1 < T_2 < T_3 < ... < T_M \leq 180$**) denoting the time of their submissions in frozen hour. **Summation of M over all test cases $\leq 1.5 \times 10^6$.**
Next line contains **N** space separated integers representing team Id from higher rank to lower rank in the leaked result (from left to right).

Output

For each test case, print the case number followed by a string "**Say no to rumour >:**" if no such result is possible, otherwise print "**We respect our judges :)**" in a single line. Check the sample I/O section for more clarity.

Sample Input

```
2
3
2 3 100 1 130
3 2 110 2 150 160
1 1 110 3 170 175 180
1 2 3
2
1 2 100 0
2 1 110 1 130
2 1
```

Output for Sample Input

```
Case 1: We respect our judges :)
Case 2: Say no to rumour >:
```

Explanation:

Case 1: A possible solution is as follows.
Team Id = 1 (4/635) , Team Id = 2 (3/100), Team Id = 3 (2/110).

Case 2: There is no way team Id = 2 can get a higher rank than team Id = 1. So, it's a rumour.

Problem D

ICPC Standings

Competitive programming is a sport that requires the elegance and the art of problem solving. ICPC, the premier global competition for programming, aims to recognize the best and the brightest young programmers from universities around the world.

Duration of an ICPC style contest is five hours. To make the contest more thrilling, the contest ranklist is frozen for the last 1 hour; that is accepted/rejected verdict is not shown in the rank list, but we can see how many times a problem is submitted by a particular team. After the contest the jury publishes the result and acknowledges the winners.

In ICPC style contest, ranklist and result is determined by the following rules:

- The team with the higher number of solved problems will get a higher rank. Rank 1 is considered as the highest rank.
- In case of a tie between two teams with the same number of problems solved, the higher rank is determined by penalty time. Team with lower penalty time will get the higher rank.
- Penalty time for a particular problem is determined by the following formula:
    - If a problem is not solved, then penalty time for that problem is **0 (zero)**.
    - Penalty for a solved problem, $\mathbf{P_A = (NS - 1) \times 20 + ST}$
      Where, **NS = Number for Submission for that problem, ST = Last Submission Time**
    - Total Penalty for a team is the summation of penalties for all problems.
      $P_{TEAM} = P_A + P_B + P_C + \dots$, where problems are named as A, B, C, … maintaining alphabetic order.
    - A team can not submit for a particular problem after getting it accepted once.
- After considering the number of solved problems and penalty time, if there is still a tie, teams will share a common rank. For example, let's assume that, both Team X and Team Y solved 2 problems with total penalty time $P_{Team\ X} = P_{Team\ Y} = 330$. It is a tie. So, they will share a common rank. In the ranklist, Team X may come before Team Y, or Team Y may come before Team X. Both are valid. The same also holds when more than 2 teams are tied.

A frozen rank list consists of **M+2** columns where **M** denotes the number of problems.
First Column: Contains Team Name and fixed column having width of twenty characters.
Second Column: Contains Number of Solve and Penalty time and fixed column having width of seven characters. It can be expressed as a fixed format **c/d**, where c is the total number of solve ($\mathbf{c \geq 1}$) and **d** is the total penalty time.
Third Column to (M+2)th column: may have any of the following formats and has fixed column width of seven characters.

- **0/-- :** This problem is not attempted.
- **c/d :** $\mathbf{1 \leq c \leq 20}$ and $\mathbf{1 \leq d \leq 240}$ means that the problem was solved in the c^{th} attempt at the d^{th} minute.
- **-c/-- :** $\mathbf{c<0}$ and $\mathbf{|c| \leq 20}$, means that the problem was submitted c times and all the submissions were rejected.
- **?c/d :** $\mathbf{1 \leq c \leq 20, 241 \leq d \leq 300}$, means that the problem was last submitted at the d^{th} minute during contest and it was their c^{th} submission on that problem. Its verdict is unknown as it was submitted in the frozen hour.

Verdicts are instantaneous and during frozen hours teams are notified only about their own submission verdict (accepted/rejected).

Team Name	Total	A	B	C	D
Team X	2/330	?10/255	0/--	1/10	5/240
Team Y	2/330	5/230	0/--	1/20	?5/270
Team Z	1/30	-4/--	1/30	?1/241	?1/245

The above table is an example of a frozen rank list, Team X had solved two problems before the rank list was frozen. They solved Problem C with penalty time P_C = (1-1)×20 + 10 =10 and Problem D with penalty time P_D = (5-1)×20 + 240 = 320. So, total penalty for Team X, $P_{Team\ X}$ = 10+320 = 330. They also submitted for problem A at 255 minutes and it was their 10^{th} submission on that problem. We don't know the verdict of 10^{th} submission as the rank list is frozen. They did not attempt problem B.

It is ICPC Dhaka Regional Preliminary Contest, 2019. The contest is already finished and the jury is ready to unfreeze the rank list and publish the result. Everything was going well. But, the judge server crashed and all of a sudden things got messy. They have lost information about which submissions were accepted and which submissions were rejected. But they have the final result consisting of one column (team names only) from higher rank to lower rank (Only the first column of the final unfrozen ranklist) .
In this problem, you have the frozen rank list and the judges' final result. You have to figure out how many different ways this final result can be achieved after unfreezing the rank list.

For example let the following be the final result of the above example contest:

```
Team Z
Team X
Team Y
```

It can be achieved in the following ways,

Team Z	2/271	-4/--	1/30	1/241	-1/--
Team X	2/330	-10/--	0/--	1/10	5/240
Team Y	2/330	5/230	0/--	1/20	-5/--

Team Z	2/275	-4/--	1/30	-1/	1/245
Team X	2/330	-10/--	0/--	1/10	5/240
Team Y	2/330	5/230	0/--	1/20	-5/--

Team Z	3/516	-4/--	1/30	1/241	1/245
Team X	2/330	-10/--	0/--	1/10	5/240
Team Y	2/330	5/230	0/--	1/20	-5/--

```
|--------------------|-------|-------|-------|-------|-------|
|Team Z              |3/516  |-4/--  |1/30   |1/241  |1/245  |
|Team X              |3/765  |10/435 |0/--   |1/10   |5/240  |
|Team Y              |2/330  |5/230  |0/--   |1/20   |-5/--  |
|--------------------|-------|-------|-------|-------|-------|
```

Input

Input begins with an integer **T** in a single line denoting the number of test cases. Each of the **T** test cases starts with two integers **N** and **M** in a single line denoting the number of teams and the number of problems respectively. Then there will be **(N+1)** $\times$ **2 + 1** lines of formatted input displaying the tabular format of the rank list. Among these

- Each odd numbered line will display the table border.
- Each even numbered line will contain M+2 columns. All these columns will be separated by '|' and they have their fixed width as mentioned in the statement above.

Then there will be a blank line.
Next N+2 lines represent the judges' final result, a single column containing team names (with the fixed width of twenty characters) from higher rank to lower rank in tabular format.

You may safely assume that:

- Table alignment will be exactly the same as it is in the sample input.
- All the values in the table will be left-aligned.
- No two teams have the same name.
- The team name consists of lowercase and uppercase Latin letters (a-z), ('A'-'Z') and white space(s) only.
- The name consists of less than 21 characters.
- At least one problem is solved by all the teams before the ranklist was frozen.

Constraints

1 $\leq$ **T** $\leq$ **50**
1 $\leq$ **N** $\leq$ **100**
1 $\leq$ **M** $\leq$ **12**

No problem is submitted more than 20 times by a particular team.

Output

For each test case, print the case number followed by the number of ways the final result can be achieved. Since the answer can be large, print the answer modulo **1000000007** (**10^9+7**).

Two ranklists RANKLIST1 and RANKLIST2 are considered different if for at least one team any of the (M+1) columns from 2nd Column to M+2th column has different values in the two ranklists.

Sample Input

```
1
3 4
|--------------------|-------|-------|-------|-------|-------|
|Team Name           |Total  |A      |B      |C      |D      |
|--------------------|-------|-------|-------|-------|-------|
|Team X              |2/330  |?10/255|0/--   |1/10   |5/240  |
|--------------------|-------|-------|-------|-------|-------|
|Team Y              |2/330  |5/230  |0/--   |1/20   |?5/270 |
|--------------------|-------|-------|-------|-------|-------|
|Team Z              |1/30   |-4/--  |1/30   |?1/241 |?1/245 |
|--------------------|-------|-------|-------|-------|-------|

|--------------------|
|Team Z              |
|Team X              |
|Team Y              |
|--------------------|
```

Output for Sample Input

```
Case 1: 4
```

Problem E

Palindrome Again? Arghh!

One day Leo was teaching Palindrome to his younger brother Neo. A palindrome is a word that reads the same backward as forward. He also showed him some palindromic words such as "**ototo**", "**madam**" and "**racecar**". Now he gave Neo the following simple problem to check whether he was inattentive during his lesson.

Neo will be given a string **s** of length **N**. The string **s** consists of only the first **K** lowercase English letters. He is allowed to make some changes to the given string. In one single operation, he can choose any two positions **u** and **v (1 ≤ u, v ≤ N)** and swap the characters at those two positions. He has to find the minimum number of operations needed to make the string palindrome.

Input

The first line contains a single integer **T (1 ≤ T ≤ 200),** which denotes the number of test cases. The first line of each test case contains two integers **N (1 ≤ N ≤ 50000)** and **K (1 ≤ K ≤ 5)**, and the second line of each test case contains the string **s**.

Output

For each test case, print a single line in the format "**Case X: R**" without the quotes where **X** denotes the case number and **R** denotes the minimum number of operations Neo needs to apply to make the string palindrome. If there's no solution then **R = -1**.

Sample Input	Output for Sample Input
5 7 4 abcdcab 8 2 abababab 5 3 abcba 6 4 abbccd 9 5 aacdedbbc	Case 1: 1 Case 2: 2 Case 3: 0 Case 4: -1 Case 5: 2

Problem F

Elevators

There are **n** elevators in a row. The elevators are numbered **1** to **n** from left to right. Each elevator has a maximum height **h[i],** which indicates the maximum height it can go upwards. Each elevator starts at time **0** from height **0**. Then they go upward at a speed **1** unit/second. After reaching height **h[i]**, they go downward at a speed **1** unit/second till they reach height **0**. Then they go upward again and the process continues. The elevators never stop after they are started.

Alice is standing on elevator **1** at time **0**. She wants to go to elevator **n**. She can jump from elevator **i** to elevator **i+1** if the current height of elevator **i** is **greater than or equal to** current height of elevator **i+1**, **and** the time is an integer number of seconds. It takes Alice **1** second to make a jump. Notice that, the height condition has to be met when Alice initiates the jump, not when the jump has been made. She can decide to wait for any number of seconds she wants before going for the jump.

Now Alice asks you when is the earliest possible time she can reach elevator **n**.

Input

The first line contains a single integer **T (1 ≤ T ≤ 5)** indicating the number of test cases. Each test case will have a single integer **n (2 ≤ n ≤ 10^5)**, denoting the number of elevators. Next line contains **n** space separated integers, **i-th** of which is the value of **h[i] (1 ≤ h[i] ≤ 10^9)** as described in the statement.

Output

For each test case, print the case number and the answer to Alice's question in a single line. Please see sample for details.

Sample Input

```
2
3
2 1 3
4
1 2 1 4
```

Output for Sample Input

```
Case 1: 2
Case 2: 8
```

Explanation of the First Sample

Height of the elevators at second **0**, **1**, and **2** are as below:

Elevator **1** - **0**, **1**, **2**
Elevator **2** - **0**, **1**, **0**
Elevator **3** - **0**, **1**, **2**

At second **0**, Alice is standing on elevator **1**. Elevator **1** is at height **0** and elevator **2** is at height **0**. So the conditions for jump is met and she makes the jump to elevator **2**. She reaches elevator **2** at second **1**. By similar logic, she can make jump from elevator **1** to elevator **2** at second **1**. She reaches elevator **2** at second **2**. This is the earliest time she can reach elevator **2**.

SSL WIRELESS

Problem G

Pairs Forming GCD

Bangladesh Association of Problem Setters

Given **N** and **P**, find maximum **G**, such that there exists at least **P** pairs of integers **(X, Y)**, where **1 ≤ X ≤ Y ≤ N** and **GCD(X, Y)** is equal to **G**.

Here **GCD(X, Y)** indicates the **Greatest Common Divisor (GCD)** of **X** and **Y** - the largest positive integer that divides both **X** and **Y**.

Input

The first line of input contains an integer **T (1 ≤ T ≤ 10^5)**, indicating the number of test cases. The following **T** lines contains two integers **N (1 ≤ N ≤ 10^7),** and **P (1 ≤ P ≤ 10^{15})** each.

Output

For each test case, print the test case number, starting from 1, and an integer, indicating the maximum possible **G**, under the given description. If there is no possible **G**, for given **N** and **P**, then print **-1** instead.

Sample Input	Output for Sample Input
4 25 12 48 125 12 60 16661 5700	Case 1: 4 Case 2: 2 Case 3: -1 Case 4: 121

Explanation of Sample I/O

In case-1, there are 12 possible pairs with G = 4.
In case-2, there are 180 possible pairs with G = 2.
In case-3, there are no possible G for which there can exist at least 60 different pairs.
In case-4, there are 5770 possible pairs with G = 121.

SSL WIRELESS

Problem H

Have You Heard of Cricket?

Bangladesh Association of Problem Setters

Have you heard of Cricket? Not the insect, we are talking about the sports! Don't worry if you haven't heard of it, because we will talk about all the rules of cricket that you need to know for this problem. Even if you know all the rules, please go through the rules again, because some of the rules may have been modified from their actual versions, for this problem.

- Two teams play against each other in a match of Cricket.
- There are two innings. In the 1st innings, one team bats, another team bowls. Then in the 2nd innings, the bowling team of the 1st innings bats, and the other team bowls.
- At first, a toss is held to decide which team will bat first.
- In each innings, 300 valid balls need to be bowled. There is an exception described in the latter points where it is not necessary to bowl 300 valid balls in an innings.
- The batting team has 10 wickets.
- Any integer value from **0 to 7** can be scored in a single ball. Also the batting team can lose a wicket in a ball. If a wicket is lost, 0 run is scored from that ball.
- Sometimes, the bowler may bowl a no-ball. A no-ball is an invalid ball. One run is added to the score of the batting team for a no-ball. The bowler has to bowl the ball again. For example, if the first ball of the innings is a no-ball and the batsman scores 4 runs from the ball, then the batting team will have 5 runs from 0 ball.
- If the batting team loses all the 10 wickets, then their innings will immediately end, even if 300 valid balls have not been bowled.
- The team batting in the 2nd innings has to score at least one run more than the score of the 1st innings to win. If they score the same as that of the 1st innings, then the match will be tied. If they score less than that of the 1st innings, then they will lose.
- The first innings will end when 300 valid balls have been bowled or the batting team loses 10 wickets.
- The second innings will end if 300 valid balls have been bowled or the batting team loses 10 wickets or their score becomes more than that of the first innings.

Team **X** and **Y** are playing in a cricket tournament. There is only one more match left in the group stage which will be held between **X** and **Y**. Only one team between **X** and **Y** will qualify to the next round. If **Y** wins the match or the match is tied, then **Y** will qualify to the next round. But even if **X** wins the match, their point will become equal to that of **Y**. So if **X** wins the match, **net run rate (NRR)** will decide which team will go to the next round. The team having the larger **NRR** will qualify to the next round. If **NRR** of both the teams is equal, then **Y** will qualify because of their current ranking. **NRR** of a team is calculated in the following way:

$$\textbf{NRR of a team} = \frac{Total\ runs\ for}{Total\ balls\ for} - \frac{Total\ runs\ against}{Total\ balls\ against}$$

For example, let's assume that team T has scored 504 runs and played 900 balls in all of its matches. In those matches, its opposition teams scored 759 runs and played 700 balls in total. Then team T's NRR will be calculated like this:

$$\textbf{NRR of team T} = \frac{504}{900} - \frac{759}{700} = \textbf{-0.5243}$$

If the batting team loses all their wickets before the completion of 300 balls, then the value of **"balls for"** of the batting team and **"balls against"** for the bowling team for that match will be **300**. If the team batting second wins the match in less than 300 balls, then the value of **"balls for"** of the batting team and **"balls against"** for the bowling team for that match will be exactly the number of balls the team batting second played.

You will be given all the data that you need to calculate the NRR of team **X** and **Y** before the last match, the result of the toss i.e. which team will bat first in the last match. You will have to find out the lowest number of runs the team batting in the **1st** innings can score after which, **X** can theoretically still qualify.

Input

In the first line, the number of test cases **T** will be given. Then for each case, three lines will be given. The first line will contain four integers signifying "**total runs for"**, "**total balls for"**, "**total runs against"** and "**total balls against"** of team **X** respectively. The second line will contain the same information for team **Y** in the same order. You can calculate the NRR of both the teams using these data. Then, the last line will contain a character **C** signifying which team will bat first.

Constraints

$1 \le T \le 3*10^5$
1 ≤ any integer in input (except for T) ≤ 10^4
C belongs to {'X', 'Y'}

Output

For each case, print the case number and then the least number of runs that the team batting first can score after which **X** can still theoretically qualify. If there is no way for **X** to qualify, print "**-1**" instead. Please see the sample for details.

Sample Input

Output for Sample Input

Sample Input	Output for Sample Input
2 504 900 759 700 10 100 1000 200 X 504 900 759 700 10 100 1000 200 Y	Case 1: 1 Case 2: 0

Note:

For the first test case of the sample input, X is batting first. Let's assume X scores 1 run in the 1st innings. Then theoretically it is possible that Y loses all 10 wickets for 0 run in the 2nd innings. If that happens, then X will win the match and they will have a better NRR than that of Y after the match. Considering the above scenario takes place in this match, the calculation of NRR of X and Y after the match are shown below:

NRR of X = $\frac{504+1}{900+300} - \frac{759+0}{700+300}$ **= -0.338**

NRR of Y = $\frac{10+0}{100+300} - \frac{1000+1}{200+300}$ **= -1.977**

Since, NRR of X is better than Y, X will qualify to the next round.

If X had scored less than 1 run, i.e. 0 run, then the match would have been either tied or won by team Y.

Problem I

Almost Forgot to Welcome

ICPC Dhaka Regional is the biggest programming competition in Bangladesh. Also the most anticipated one as well. Students from all the different universities storm their brains all year in preparation for this competition. You are now taking part in this competition, so a big congratulations to you.

Dhaka site is one of the biggest sites in ICPC. This year almost 1800 teams are participating in the competition. So in celebration, we are going to give you an easy problem to solve.

You have to write a program, which will print the line “Welcome to ICPC Dhaka Regional Online Preliminary Contest, 2019” (without quotes).

Note: you can’t output anything other than the required output, and the line must end with a newline (‘\n’). Take special care about spelling and case. If you alter any of those, you may not get accepted.

For your convenience, we are providing one sample program in C/C++ which prints “Bangladesh”. You just have to change the code to your requirement.

```
#include <stdio.h>
int main()
{
    printf("Bangladesh\n");
    return 0;
}
```

SSL WIRELESS

Problem J

Good Things Come to Those Who Wait

Bangladesh Association of Problem Setters

Finding divisors of an integer is common task. We all did it in some time or other during our programming courses. We know that there are several ways to find divisors of an integer **N**. All of them involve some kind of programming using loops. But what if we ask you to solve the reverse problem?

Given all the proper (all the divisors except **1** and **N**) positive divisors of a positive composite integer **N**, you need to find the value of **N**.

Input

The first line of the input contains an integer **T (0 < T < 11)**, denoting the number of test cases. The next **T** lines will contain the test cases. For each test case an integer **K (K > 0)** will be given which is the number of proper positive divisors of **N**. The following line will consist of **K** integers. These are all the proper positive divisors of **N**. You should assume that the divisors will be given in such a way so that the value of **N** will always be greater than 1 and less than or equal to 10^{12}.

Output

For each test case, print a single line in the format **"Case X: N"** (without the quotes) where X denotes the case number and **N** is the answer. Refer to the sample input/output for more clarity on the format.

Sample Input	Output for Sample Input
3 2 2 4 1 3 2 5 2	Case 1: 8 Case 2: 9 Case 3: 10

RUET CSE FEST 2k22
Inter University Programming Contest

Technical Partner:

04 June, 2022

You get 15 Pages, 11 Problems & 300 Minutes

Problem A

Maiden Over

We all know about cricket, a game which is played between two teams. Each team bowls and bats alternatively. The team that scores more runs, is declared the winner. The aim of the batsmen is to score runs while the aim of bowlers is to minimize the run of the opposite team.

The person who bowls to the batters is known as a bowler. A bowler bowls six balls consecutively to finish an over. After finishing an over another bowler starts to bowl from the other end of the pitch. If a bowler doesn't give any run in a particular over, then, that over is considered a maiden over.

Mr. X is one of the deadliest bowlers nowadays. He doesn't get wickets because batsmen are very aware of his bowl and play carefully. In this problem, you are given how many runs were scored in each bowl of Mr. X. You have to determine how many maiden overs were bowled.

You may safely assume that Mr. X does not bowl any no ball or wide ball and for batters, there are no other ways to score runs without hitting a ball.

Input

The first line will contain **T (1 ≤ T ≤ 1000)**, the number of test cases. The following **T** test cases will start with an integer **O (1 ≤ O ≤ 10),** denoting the number of overs bowled by Mr. X. Each of the next **O** lines will consist of **6** integers describing the runs scored in each ball.

Output

For each case, print one line with **"Case <x>: <y>"**, where **x** is the case number and **y** is the number of maiden overs bowled. Please see the sample output for more details.

Sample Input	Output for Sample Input
2 1 0 0 0 0 0 0 1 1 0 0 0 0 0	Case 1: 1 Case 2: 0

Problem B

All About Constraints

In this problem, all you have to do is generate an array, having **N positive** integers, which satisfies the given **Q** constraints. Every constraint is of the following kind - "The bitwise **XOR** of all the numbers from the **L**-th position to the **R**-th position of the array has to be equal to the **SUM** of all the numbers from the **L**-th position to the **R**-th position". You can only use numbers from 2^0 to 2^{63} You need to find the lexicographically smallest array which satisfies all the constraints or claim that such an array does not exist.

Input

The first line of the input contains an integer **T (1 ≤ T ≤ 10)**, denoting the number of test cases. Then the description of the **T** test cases follows. The first line of every test case will contain two integers **N** and **Q**, (**1 ≤ N, Q ≤ 10^5)** denoting the length of the array and the number of constraints respectively. Each of the following **Q** lines will contain a constraint of the following form: **L R** (**1 ≤ L ≤ R ≤ N)**.

Output

For each test case, print a line containing the case number, and if such an array can be constructed then print the lexicographically smallest array which satisfies all the given constraints. Otherwise, print "**Impossible**". Please make sure two consecutive elements in the array are separated by a single space. And there are no extra spaces after the last element. Please see the sample output for more details.

Sample Input	Output for Sample Input
2 6 2 2 3 6 6 100000 1 1 100000	Case 1: 1 1 2 1 1 1 Case 2: Impossible

Note: An array **C** of size **N** is called lexicographically smaller than another array **D** of equal size, if and only if there exists an **i ≤ N** such that: **C[p] == D[p]** for all **1 ≤ p < i** and **C[i] < D[i]**

Problem C

Some Game

Alfonso and Bethany are playing a game. In this game, there are **P** boxes numbered from **1** to **P**. The i-th box has **A[i]** balls initially. All of the balls in the game are distinct. The boxes are similar, i.e. there is no way to distinguish one box from the other.

At first, the players are given a target integer **T** (the significance of **T** is explained towards the end). The game consists of **P** sequential moves. Alfonso gives the first move and then Bethany gives the 2nd move, and so on. So if **P** is odd, Alfonso gets one extra move, but Bethany is okay with it.

In the i-th move of the game, a player simply decides if they want to destroy the i-th box or not. After **P** moves, **K** boxes will remain where **K** is the number of moves where a player had decided not to destroy a box. Let **N** be the total number of balls in these **K** boxes. Then the players take out these **N** balls from these **K** boxes and calculate the following value for **R**:

> ***R** = (The number of ways to put **N distinct** balls in a linearly ordered way in **K** **indistinguishable** boxes so that none of the boxes remain empty) mod 10^9+7*

See the notes section for an example of how **R** is calculated.

Alfonso's target is to give moves in such a way so that ***abs(T - R)*** is **maximized**. Bethany's target is to **minimize *abs(T - R)*.** *abs(x)* denotes the absolute value of x.

Note that we care about the value of **R** after doing the mod operations.

If both play optimally, what will be the value of ***abs(T - R)***?

Input

The first line will contain a single integer **Q**, denoting the number of test cases. The first line of each test case will have two integers **P**, denoting the number of boxes, and **T**, the target integer. The next line will contain **P** integers where the i-th integer denotes **A[i],** the number of balls in the i-th box.

Constraints

- **$1 \le Q \le 5$**
- **$1 \le P \le 30$**
- **$1 \le A[i] \le 10^6$**
- **$1 \le T \le 10^9$**

T and all A[i] are generated using a pseudo-random generator. The generator is used to make the inputs random and ensure inputs are not adversarially generated against any particular solution.

Output

Print the case number in a single line followed by the value of **abs(T - R)**. Please see the sample for details.

Sample Input	Output for Sample Input
3 1 100 10 2 19490 1 2 5 123456789 5555 1000000 1 342 1653	Case 1: 3628700 Case 2: 19488 Case 3: 687865730

Notes

Suppose, we have 3 balls A, B, and C and 2 boxes.

- {A} {BC} and {A} {CB} are two different ways. (As the linear order of balls inside a box matters)
- But {BC} {A} and {A} {BC} are the same ways (As the boxes are similar)
- {} {ABC} is not a valid way since no box can be empty.

For N = 3, K = 2, the value of R is (6 % (10^9+7)) = 6

.

Explanation of sample 1:

In this game, Alfonso has one move and Bethany has 0 moves.

- Alfonso can destroy the first box making R = 1 and getting ans = abs(100 - 1) = 99.
- Alfonso can decide to keep the first box making R = (10! % (10^9+7)) = 3628800 and getting answer = abs(100 - 3628800) = 3628700. Since Alfonso wants to maximize the final answer, he will decide to keep the box.

Problem D

Maxxxxxximum Spanning Tree

You are given a complete graph of size **n** where the nodes are labeled from 1 to **n**. The weight of the edge between any two nodes **i** and node **j** is equal to the gcd(**i**, **j**). Find the cost of the **maximum** spanning tree for this graph.

Input

The first line contains an integer **T**, denoting the number of test cases. Then **T** test cases follow. Each test case contains a single integer **n** in a separate line.

Constraints

- **$1 \leq T \leq 10$**
- **$1 \leq n \leq 1000000$**

Output

For each test, print one line in the format "**Case X: Y**", where **X** is the case number and **Y** is the cost of the maximum spanning tree for that case.

Sample Input	Output for Sample Input
2 **1** **4**	**Case 1: 0** **Case 2: 4**

Problem E

Number of Zeros

Hermione is once again trying to solve the following problem in arithmancy class:

Let us consider an array of **N** elements $V = [v_1, v_2, \dots, v_n]$. Now we do the following operations repeatedly on V till there is only one element in **V**:

1. $W = [v_1 \times v_2, v_2 \times v_3, \dots, v_{n-1} \times v_n]$
2. $V = W$

Now if initially $v_k = k!$ and then we continue this operation till there is only one element in **V**, what would be the number of trailing zeros in that number in base **B**?

As a friend of Hermione, your task is to solve this problem by writing a program.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 200),** the number of test cases. Then T cases follow. Each test case will have two integers **N (1 ≤ N ≤ 10^5)** and **B (2 ≤ B < 10^5)** in a separate line. **B** will always be a prime number.

Output

For each case, print the case number in a single line followed by the number of zeros. Please see the sample for details. As the number of trailing zeros can be too large, print the **number modulo 1,000,000,007 (10^9 + 7)**.

Sample Input	Output for Sample Input
3 5 5 3 3 4 5	Case 1: 1 Case 2: 1 Case 3: 0

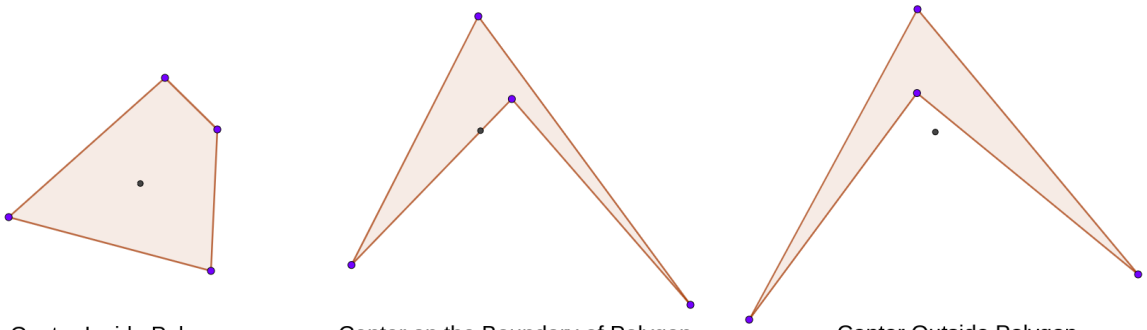
Explanation of Sample Input 2: Given **N = 3, V = [1, 2, 6]**. Then after 1st iteration, **V = [2, 12]** and after 2nd iteration, **V = [24]**. If we write it in base **3**, it is **220,** where there is 1 trailing 0.

Problem F

Unbalanced Polygon

The center of a polygon is the average of its vertices. Formally, the center of the polygon with **n** vertices $(x_1, y_1), (x_2, y_2), (x_3, y_3) \ldots\ldots (x_n, y_n)$ is $\left(\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i\right)$

A polygon is called unbalanced if it is simple and the center of the polygon lies strictly outside the polygon.



Center Inside Polygon

Balanced

Center on the Boundary of Polygon

Balanced

Center Outside Polygon

Unbalanced

Alice has a convex polygon with **n** vertices, $P = P_1P_2P_3 \cdots P_n$ For each vertex $\mathbf{P_i}$, Alice tries to move the point $\mathbf{P_i}$ to a new point **S** anywhere **inside** the polygon **P**, such that the resulting polygon is unbalanced. For each vertex $\mathbf{P_i}$, help Alice find out the area all such points cover.

Formally, Let $T(i)$ be the set of all points **S** inside **P**, such that $P = P_1P_2 \cdots P_{i-1}SP_{i+1} \cdots P_n$ is an unbalanced Polygon. Find $Area(T(i))$ for all $1 \leq i \leq n$. It can be proven that the area is finite.

Input

The first line contains an integer **T (1 ≤ T ≤ 50)** denoting the number of test cases.

The first line of each case contains an integer **n (3 ≤ n ≤ 5000)** the number of vertices of the polygon **P**.

Each of the next **n** lines contains two integers $\mathbf{x_i}$ and $\mathbf{y_i}$ **($-10^5 \leq x_i, y_i \leq 10^5$)** - the coordinates of $\mathbf{P_i}$.

The vertices of the polygon are given in counter-clockwise order. It is guaranteed that the vertices form a convex polygon.

Output

For each test case, output **n** space-separated real numbers. The **i**-th number should be **Area(T(i))**, the answer for the **i**-th point. Your answer will be considered correct if its relative or absolute error doesn't exceed 10^{-4}.

Sample Input	Output for Sample Input
2 3 0 0 1 1 1 2 4 0 0 1 0 1 1 0 1	0.0 0.0 0.0 0.166667 0.166667 0.166667 0.166667

Problem G

Polymorphism

Polymorphism is one of the four main pillars of Object Oriented Programming. It means "many forms", and it occurs when we have many classes that are related to each other by inheritance. **Inheritance** lets us inherit attributes and methods from another class.

In this problem, we will only focus on creating classes and objects by following these properties. We are designing a system that will support two types of operation, either we will add a new class in the system, or ask whether we can create an object at this state or not. Details are described below.

Creating classes:
We already have an existing class in our system called **Main**. All the other classes will be inherited from one of the already declared classes. Class names are case-sensitive and unique. A plus sign ("+") will be used while creating a new class. It will follow this format

```
+ class SubClassName extends SuperClassName
```

Here, **`SubClassName`** is the newly created class and **`SuperClassName`** is the parent class it inherits from. And **`extends`** is the keyword that denotes the inheritance. That Superclass has to be created before declaring this new Subclass. We assure you that all the classes in the input are created in a valid way.

Creating objects:
We can create objects from the existing classes in two ways. This will contain a question ('?') sign in the beginning

```
a)    ? ClassName objectName = new ClassName()
b)    ? AncestorClassName objectName = new ClassName()
```

To create an object, specify the class name, followed by the object name, and use the keyword **`new`**. In the second method, there has to be a chain of inheritance from **`AncestorClassName`** to **`ClassName`**. Object names are case-sensitive and unique as well.

Here's an example for better understanding. Suppose we have created these classes.

```
+ class Animal extends Main
+ class Dog extends Animal
+ class Cat extends Animal
+ class Puppy extends Dog
```

Now some valid object creations are:

```
Animal animal = new Animal()
Main abcd = new Animal().
```

`Main xyz = new Puppy()` - This is valid because *Main* is an ancestor class of *Puppy* class.
`Animal catAnimal = new Cat()`

Some invalid object creations are:
`Animal object = new Main()` - This is invalid, because *Animal* is not an ancestor class of *Main*.
`Main myObject = new MyClass()` - This is invalid because we don't have any class named *MyClass*.

Now in this problem, you have to determine whether each object creation is valid or not.

Input

Input starts with an integer **T,** number of test cases. The first line of each test case starts with integer **N**, which indicates how many lines to follow. Then each of the next **N** lines starts with either a plus sign (+) which means a new class is created or a question sign (?) which asks a query whether this object creation is valid or not.

Constraints

- **1 ≤ T ≤ 100**
- **1 ≤ N ≤ 100000**
- **1 ≤ Length of each class name ≤ 10**
- **1 ≤ Length of each object name ≤ 10**

Summation of all **N** in a test file will be less than or equal to **200000**.
Class and object names consist of lower-case, upper-case characters, and numeric digits.
All names and keywords are case-sensitive and unique.

Output

Print the case number in a single line. Then for every query, you need to print "yes" only if it is a valid way of creating an object or print "no" otherwise. Please see the sample for details.

Sample Input	Output for Sample Input
1 7 + class Abc extends Main ? Abc obj = new Abc() ? Main obj2 = new Abc() ? Abc obj3 = new Main() + class Xyz extends Abc ? Main obj4 = new Xyz() ? Abc obj5 = new Xyz()	Case 1: yes yes no yes yes

Note: Dataset is huge. Please use faster input/output methods.

Problem H

Digit Shifts

You will be given a **non-negative integer X**. You need to perform **Q** queries on that integer. Each query will consist of a single decimal digit **D**. After every query, you need to move all occurrences of digit **D** in the integer to the end, while keeping the relative position of every other digits intact.

For example, suppose **X = 123123,** and suppose **Q = 3**.

1. For the first query, **D = 1**, then after the digits are shifted **X = 232311**.
2. For the second query, **D = 2**, then after the digits are shifted **X = 331122**.
3. For the third query, **D = 3**, then after the digits are shifted **X = 112233**.

After every query, you need to output the value of the integer **X**. Since it can be really large, output it modulo **1000000007** (**10^9 + 7)**.

Please note that if at any point after a query, **X** contains leading zeros, then the leading zeros should be **discarded**. Therefore, if **X = 2022** and if **D = 2**, then after the query, **X** will become **222**.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 20)**. Each starts with a single line, which will contain the **integer X**. Then, in the next line there is a single integer **Q (1 ≤ Q ≤ 10^5)**, denoting the number of queries. Each of the next **Q** lines denotes a query, containining a decimal digit **D (0 ≤ D ≤ 9)**. You can safely assume that **X** won't contain leading zeros initially and that **X** will never have more than **10^5** digits.

Output

For each test case, first print the case number in a single line like "**Case V:**". Then for each query, output the value of **X** modulo **1000000007**. Refer to the sample I/O for more clarity.

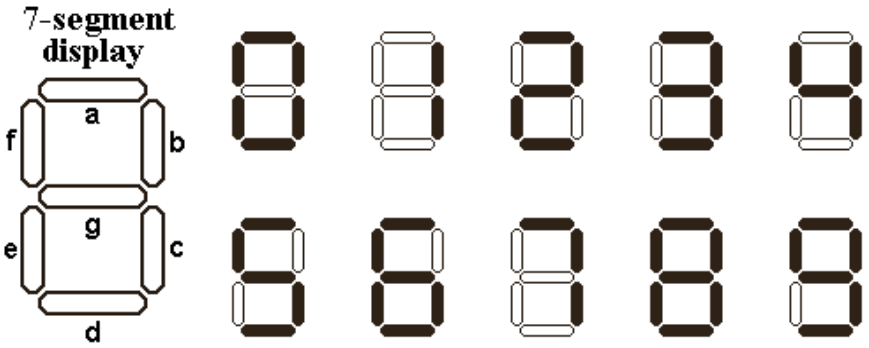
Sample Input	Output for Sample Input
1 123123 3 1 2 3	Case 1: 232311 331122 112233

Note: Dataset is huge. Please use faster input/output methods.

Problem I

Seven Segment Display

You are given an integer. The integer is written using sticks in the same format as 7 segment display. The figure below shows the digits (0 - 9) in seven segment display format.



In the given integer, you can't add or remove any sticks. But you can move some (possibly none) sticks. In a single move, you can take one stick from one position and put it to a different position which is empty. The move can happen from one digit to another digit. It can also happen between the same digit as well. The length of the starting number and the final number must be the same. What is the maximum integer you can make by using at most **K** moves?

Input

The first line will contain a single integer **T (1 ≤ T ≤ 20)**. Each test case will have a two integers **S, Q (1 ≤ Length of S, Q ≤ 200)**, where **S** is the given integer in seven segment display and **Q** is the number of queries. **Q** lines will follow. **S** will be a non-negative integer, with possible leading zeroes. Each line will contain an integer **K** (**1 ≤ K ≤ 200)** which denotes the number of moves. For each **K**, you will need to produce the maximum integer you can make from **S** by using at most **K** moves.

Output

Print the case number in a single line followed by query answers in a single line. Please see the sample for details.

Sample Input	Output for Sample Input
3 123 3 1 2 3 011 2 1 2 111 2 1 100	Case 1: 133 724 797 Case 2: 911 911 Case 3: 111 111

Problem J

K Subsequence Problem

There is an array of **N** integers. How many subsequences of it have at least **K** distinct numbers?

Input

The first line of the input contains **T(1 ≤ T ≤ 20)**, the number of test cases. The following **T** tests each contains two lines. The first line has two numbers **N** and **K(1 ≤ K ≤ N ≤ 100000)**. The next line contains **N** integers in range **[1..N]**.

Output

For each test print one line in the format "**Case X: Y**", where **X** is the case number and **Y** is the number of subsequences. Since the result can be very large, print it modulo **998244353**.

Sample Input	Output for Sample Input
3 3 2 1 2 3 3 1 1 1 2 3 2 1 1 2	Case 1: 4 Case 2: 7 Case 3: 3

Problem K

Change the usernames

One of the most popular online judges of Bangladesh, CodeMania (also known as CM) has recently introduced a new feature - now users can change their username if the new username they want is not being used by anyone else.
For example, let's say the user with a username **"theredwarrior"** wants to change it to **"thegreenwarrior"**. If no one is using it then the user should be able to change it to the new one. As soon as the change happens, the old username "theredwarrior" will become available for anyone to use. Users can request to change their username as many times as they want. But for a particular request, if the new username isn't available, then that operation will fail.

Suddenly the developers of CM are in a situation where they need to know what username an user initially had before the new feature was introduced. From the example above, the developers would like to know what username **“thegreenwarrior”** had before the new feature was in place. In that case, the answer should be **“theredwarrior”**.
The developers are in hurry and unfortunately, they are unable to find any optimized solution for it. So, they are asking you to help them out.

Input

The first line will contain **T (1 ≤ T ≤ 10)**, the number of test cases. Each case will start with **N (1 ≤ N ≤ 100000),** the number of instructions the developers give to you. Each of the instructions can be one of the following two types:

1 X Y: This is the change instruction, username **X** wants to change to username **Y**. **X** is a username which someone is using at this time. Y is the new username that the user wants. **X** and **Y** will always be different.
2 X: For the user who currently has username **X** the developers want to know what the username user initially had before the new feature took place.

In all the instructions, the given usernames will consist of only lowercase letters, and the length of the usernames will be at most **20**. The input file size will never exceed **10MB**.

Output

For each of the instructions of type 2, print the initial username the user had before the new feature took place. If the username was released by any earlier instructions provided by the developers and no one is currently using it, then just print **“Not in use!”** (without the quotes).

Sample Input	Output for Sample Input
1 6 1 theredwarrior thegreenwarrior 1 theyellowwarrior thegreenwarrior 2 thegreenwarrior 2 theyellowwarrior 2 theredwarrior 2 theorangewarrior	theredwarrior theyellowwarrior Not in use! theorangewarrior

Explanation of Sample Input:

- **1 theredwarrior thegreenwarrior** (the username of **theredwarrior** is changed to **thegreenwarrior**)
- **1 theyellowwarrior thegreenwarrior** (instruction failed, **thegreenwarrior** already in use)
- **2 thegreenwarrior** (Initial name of **thegreenwarrior** was **theredwarrior**)
- **2 theyellowwarrior** (Initial name of **theyellowwarrior** is **theyellowwarrior.** In past instructions we can see **theyellowwarrior** wanted to change the user name which was unsuccessful.)
- **2 theredwarrior** (No one is currently using it)
- **2 theorangewarrior** (**theorangewarrior** was never changed, so initial username of **theorangewarrior** is **theorangewarrior**)

Problem A (Maiden Over)

Setter: Mohammad Ashraful Islam
Tester:
Alter: S.M. Shaheen Shah
Category: Adhoc

Problem B (All About Constraints)

Setter: Md Shafiul Islam
Tester: Mohammad Solaiman
Alter:
Category: Greedy, Construction

Editorial:

It is always optimal to use numbers which are powers of 2. There are only 64 such numbers. So if a range's length is greater than 64 then there is no answer.

Now we will greedily put numbers in ascending order as we have to find the lexicographically smallest array. For that, we will sort the constraints by smaller L.

Assuming there is three ranges [2 7], [3 6], [3 10] and N = 10

At first for position 2 to 7 we will put [1, 2, 4, 8, 16, 32]

So our array looks like this

_, 1 , 2 , 4 , 8 , 16 , 32 , _ , _ , _ [_ represents no number yet]

Next comes the [3 6], As all the position is already covered by [3 6], we can safely ignore this constraint as it is already satisfied.

For the last range position 4-7 is already has a number and we will keep it that way and put 1 64 128 in position 8-7. We cannot use 2, 4, 8, 16, 32 again. And put 1 in the blank positions. So final answer is

1 , 1, 2, 4, 8, 16, 32, 1, 64, 128

Problem C (Some Game)

Setter: Tanzir Pial
Tester: Rafid Bin Mostofa
Alter: Ashiqul Islam
Category: Alpha-Beta Pruning, Combinatorics

Let F(n,c) be the number of ways n distinct balls that can be linearly arranged in k indistinguishable boxes.

F(n,c) = nCr(n-1, c-1) * n! / c!

This can be found out using techniques similar to stars and bars analogy. We can place the n balls in n! Different ways and there will be n-1 gaps between them. From these n-1 gaps, we pick c-1 gaps in nCr(n-1,c-1) ways for placing bars to denote c non-empty boxes. Finally we need to divide by c! Since we are overcounting the arrangement of the boxes.

Now that we know a O(1) formula for solving F(n,c), all that remains is the backtrack solution for the minimax game. A naive brute force will be O (2^P). Due to the random nature of the input, a backtrack with alpha beta pruning will be average case O(2^(.75P)). *

F(n,c) has a O(n*c) dp solution too but that will require too much memory and time.

Bonus: Is it possible to create adversarial input for a solution using alpha beta pruning for this problem?

* In general, for a backtrack with b branching factor and d depth, alpha beta pruning brings the complexity down to O(b^(.75d)) average case runtime for random values at leaf nodes.

Problem D (Maxxxxxximum Spanning Tree)

Setter: Rezwan Mahmud
Tester: Rafid Bin Mostofa
Alter:
Category: Graph, Number Theory
Only keep those edges(i, j) where j is a multiple of i. You will have O(n ln n) edges in total for consideration. All the other edges can be discarded(Proof is left as an exercise to the reader). Now you can run Kruskal's algorithm. Note that you have to do counting sort to sort the edges in order to fit your solution under the TL.

Problem E (Number of Zeros)

Setter: Anindya Das
Tester: Hasnain Heickal
Alter: Aminul Haq
Category: Number theory, Combinatorics

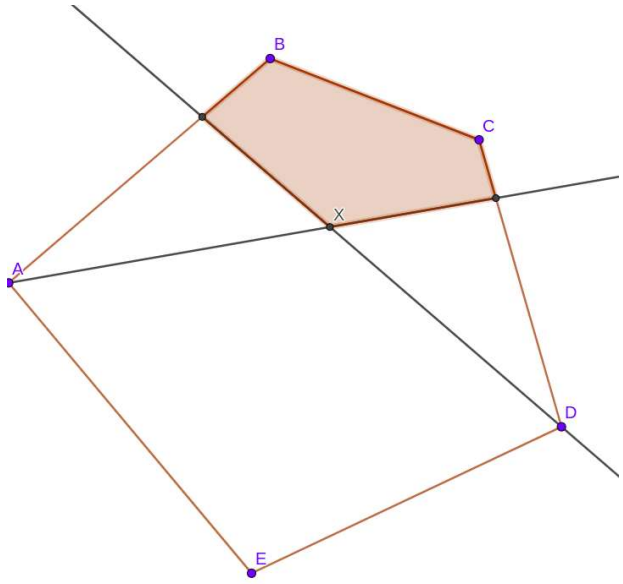
First we need to find the formula for V in terms of ai. We can observe that it would be the product of ai ^ (n-1)C(i-1). Now you just need to find out the number of zeros in factorial(i) and multiply it by (n-1)C(i-1) (use the formula: nCr = n! / (r! * (n-r)!), precalc all factorials and their modular inverses upto 10^5) and sum it over all i.

Problem F (Unbalanced Polygon)

Setter: Pritom Kundu
Tester: Muhammad Ridowan
Alter: Nafis Sadique

Tags: Geometry

First, we will find a characterization for T(i). Let X be the center of all vertices except P_i. Then T(i) is the area of the region of the polygon bounded by the two lines $P_{i-1}X$ and $P_{i+1}X$. See the figure for clarity. The proof is left as an exercise to the reader.



Region of Unbalanced positions for point E.

After this characterization is found, the task left is to calculate this area for each vertex efficiently.

First we find which side of the polygon $P_{i-1}X$ intersects with. Note that the line divides the polygon into two regions. The intersecting side is the one, for which the two endpoints are on opposite sides of the polygon. Thus we can find the furthest point on the left side with a binary search. Then we can easily find the intersection of $P_{i-1}X$ and this side. Let's call this point Y.

Similarly, we find which side of the polygon $P_{i+1}X$ intersects with. Then we can easily find the intersection of $P_{i+1}X$ and this side. Let's call this point Z. This part takes O(log n) time per vertex.

Then we can have determined all vertices of the region and can calculate the area. Note that each region has at most four sides that are not also sides of the given polygon. On the other hand, it can be proven that each polygon side can be strictly inside at most two regions. The two facts together prove that there are no more than 6n sides over all n regions. Thus the area calculation can be done in brute force in O(n) amortised time.

Alternatively, a prefix-sum like technique can be used to calculate the area in O(1) time per vertex. Overall time: O(n log n)

Problem G (Polymorphism)

Setter: Aminul Haq
Tester: S. M. Saheen Sha
Alter: Imran Bin Azad
Category: DFS, Sparse table
Solution: First, let's understand the problem, it says you have a root node of a tree which is the "Main" class and then in each query you either add a new node to the tree or answer whether a given node is an ancestor of another given node or not.
There are two soutions, one is to find LCA using sparse table and the other one is to solve the problem offline with a simple DFS. Let's discuss the second one. For this, first, we will read all the queries, and build the tree, and remember "Main" class is the root of the tree. Then we can start a DFS from from the root and keep track of all the ancestor nodes, for that we can use a Set. And then when we visit a node, then we can check all the queries on it.
Complexity: O(N + Q) where N is the number of nodes, and Q is the number of queries.

Problem H (Shift Digit)

Setter: Sabit Zahin
Tester: Rafid Bin Mostafa
Alter: S.M. Shaheen Sha
Tags: Ad hoc, DS, Simulation
Solution:
There are two key points to solve this problem.
1. Number of Digits is Only 10
2. If a digit shifts, that will form a group of the same digits and will stay together for the remaining operations.

Now, coming to the solution. To solve this Problem, we will maintain two lists. First list will contain digits in the given order. When we shift a digit we will move this digit to the end of the Second List, and delete that digit from the first list, the rest of digits will remain the same with the given order. Now think, In the second list, do we need to store all the digits ? No. We will store only that digit and frequency. If the digit is already in the second list, we will change the order only in that list. After every operation we have to truncate the leading zeros if any.
Now calculating the value, now observe that the first list will change at most 10 times. So we can calculate the value for this list by iterating all indexes after each change in that list. Suppose there are M digits (d1,d2,.. dm) in the first list and S = d1 * 10^(M-1) + d2 * 10^(M-2) + … +dm^ 10^0

For the second list, suppose there are N digits (x1, x2, …. xn) and these frequencies are (f1,f2,..fn)
We can pre-calculated this,
S1 = x1*10^0 + x1*10^1+....x1*10^(f1-1)
S2 = x2*10^0 + x2*10^1+....x2*10^(f2-1)
…
Sn = xn*10^0 + xn*10^1+....xn*10^(fn-1)
Then Final value will be, = S*10^(f1+f2+...+fn) + S1*10^(f2+f3+..+fn) + … + Sn

Problem I (Seven Segment Display)

Setter: Hasnain Heickal
Tester: Md Mahamudur Rahaman Sajib
Alter: Muhiminul Islam Osim, Abdullah Al Maruf
Tags: Dynamic Programming

Suppose two numbers S and T are given. We need to make T from S using minimum number of moves. Let's try to solve this problem first.
a = number of sticks which are at the same position in both T and S.
b = number of sticks which are available in T but not in S.
c = number of sticks which are available in S but not in T.
It is very easy to see that b = c and we can say that we are moving only c number of sticks. So the minimum number of moves is b = c.
Now let's try to solve another problem. Given S, we need to make a number T such that the minimum number of moves are minimised. We just need to find that minimised minimum number of moves. Let's try to solve the problem using dynamic programming and we will use this dp information to solve the actual problem.
dp[i][j] = minimum number of sticks which are in T but not in S to cover [i : n - 1] position using j number of sticks(as we said above minimum number of moves is b = c, so we are trying to minimise b).
cost(S[i], d) = number of sticks which are available in digit d but not in digit S[i].
dp[i][j] = min(cost(S[i], d) + dp[i + 1][j - stick_count(d)]) [0 <= d <= 9]
Now we can use this dp table to solve the actual problem as dp[i][j] is equivalent to the minimum number of moves.
In this problem we need to build a maximum number using k number of moves. So let's try to solve this problem greedily. We will try to put the maximum digit in the most significant position. For example we already filled up [0 : i - 1] positions correctly and we used x number of sticks and y(it can be easily found with the above theory) number of moves already. Now we want to put a digit in the i'th position so that the number is maximised. We will try all the digits, d from 9 to 0 and check the dp[i + 1][total_stick - x - stick_count(d)]. If y + cost(S[i], d) + dp[i + 1][total_stick - x - stick_count(d)] <= k, then we can say that digit d is a valid digit. We will put the maximum valid digit. So time complexity of building the dp is O(n * n * 10) per test case and time complexity of query solving part is O(q * n * 10) per test case. Total time complexity O(t * (n * n * 10 + q * n * 10)).

Problem J (K Subsequence Problem)

Setter: Md. Nafis Sadique
Tester: Sabit Zahin
Alter: Pritom Kundu
Tags: Combinatorics, FFT, Divide and Conquer

Let's think about each unique integer that occurs in the array. If it occurs c times, then we can include at least one occurrence of it in our subsequence $2^c - 1$ ways.

Let $a_1, a_2, a_3, \cdots, a_p$ be the unique elements in the array. Let c_i be the number of time a_i occurs in the array. We define the polynomials $P_i := 1 + (2^{c_i} - 1)x$. Consider the product of these polynomials $P = P_1 P_2 \cdots P_p$. The coefficient of x^k in this polynomial is our required answer.

Note that, P has degree at most n. Thus the coefficients of P can be calculated by divide and conquer and polynomial multiplication (with fft). Overall complexity: $O(n \log^2 n)$

Problem K (Change the username)

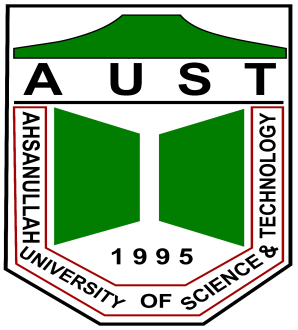
Setter: Raihat Zaman Neloy
Tester: Nafis Sadique
Alter: S. M. Shaheen Sha
Tags: AdHoc, STL (map/set)

This problem is solvable using two maps or one map and a set. Whenever the first operation appears, we just need to check if a user is already using the username Y. Then the change operation won't happen and at the same time we will store the information of X into our username database. If the change operation is successful, then we will delete info of X from our username database and keep X in another map or set which let us know about the usernames that are released because of operation 1. One thing we should keep in mind, when username X is released it will behave like a totally new username in future if ever used.

For operation two, we need to loop up for the username X if we have ever seen that before. If not, then we will print the X as output and at the same time store it's information in the database. Otherwise if we find X in released database information, we will print "Not in Use!", otherwise the expected answer we were keeping track of.

Cefalo CodeFiesta 2022
AUST Inter University Programming Contest

Brought to you by



In association with

Event Sponsor

25 June, 2022

You get 14 Pages, 10 Problems & 300 Minutes

Problem A

K^{th} Permutation Revisited

A sequence of **N** integers is called a permutation if it contains all integers from **1 to N** exactly once. For example: if **N=3**, then the set is **{1,2,3}** and it has a total **3! (= 6)** permutations.
By listing and labeling, all of the permutations in order are shown below.

1. **{1, 2, 3}**
2. **{1, 3, 2}**
3. **{2, 1, 3}**
4. **{2, 3, 1}**
5. **{3, 1, 2}**
6. **{3, 2, 1}**

In this problem, you are given **N**, **K** and **M**. You have to find the summation of the last **M** integers (rightmost) of the **K^{th}** permutation of **N** integers.

For example, if **N=3, K=2 and M=2**, then the **K^{th}** Permutation is **{1, 3, 2}**. So, the answer should be **3+2=5**.

Input

The first line will contain **T (1 ≤ T ≤ 100000)**, the number of test cases. Each test case consists of one line containing three integers **N (1 ≤ N ≤ 10^8), K (1 ≤ K ≤ min(10 , N!)) and M (1 ≤ M ≤ N).**

Output

For each case, print one line with "**Case <x>: <y>**", where **x** is the case number and **y** is the summation of last **M** integers (rightmost) of the **K^{th}** permutation of the sequence consisting of **N** integers. Please see the sample IO for more details.

Sample Input	Output for Sample Input
4 3 2 1 3 2 2 4 1 4 4 1 2	Case 1: 2 Case 2: 5 Case 3: 10 Case 4: 7

Problem B

Yet Another Suffix Array Problem

Stelle has recently learned the Suffix Array algorithm. But she got upset realizing that problems which require Suffix Array rarely come up in Bangladeshi contests! So, the problem setter has decided to develop a problem and give it to her to solve it!

A sequence of N integers is called a permutation if it contains all integers from 1 to N exactly once. The problem setter will give Stelle a permutation P. Then he will ask her **Q** queries, each being like this: "What is the lexicographically smallest **K**th subarray of the permutation **P**?"

Input

The first line will contain **T (1 ≤ T ≤ 10)**, the number of test cases. The first line of each test case will contain two integers **N (1 ≤ N ≤ 100000)** and **Q (1 ≤ Q ≤ 100000),** the size of the array, and the number of queries respectively. The next line contains **N** space-separated integers, denoting the elements of the array. The next line contains **Q** space-separated integers, each denoting the query **K (**1 ≤ K ≤ $\frac{N\times(N+1)}{2}$ **)**.

Output

For each of the queries, print a line containing two space-separated integers **L R** such that **1 ≤ L ≤ R ≤ N** and **P[L…R]** is the **K**th lexicographically smallest subarray of **P**.

Sample Input	Output for Sample Input
1 3 2 2 3 1 1 6	3 3 2 3

The subarrays of [2 3 1] sorted lexicographically smallest to largest are:
[1]
[2]
[2 3]
[2 3 1]
[3]
[3 1]

Problem C

Connect the Cities

The King of Merryland recently conquered a lot of cities to show the strength of his empire. But the newly conquered cities have no roads between them to easily communicate. As a result, it will be hard for the King to rule and develop the business. So, he asked one of his ministers to build bidirectional roads among them but he also gave the following conditions:

- After the roads are built, all the cities should be easily accessible. From any city, they should be able to travel anywhere using **at most two roads**. A single road connects two cities. Also, there will be only a single way to reach from one city to another.
- As the king is also concerned about finance, he wants the total cost of building the roads to be minimum. Each proposed road has a cost associated with it.

The minister has all the manpower to build the roads and he has also prepared a list of roads that can be built. But he has no idea how to choose the roads from that proposed list so that the cost will be minimal. Someone told the minister about you and your friends who are experts in programming and solving such problems. So, the minister is now asking you to help him out.

Input

The first line will contain **T (1 ≤ T ≤ 10)**, the number of test cases. Each case will start with two integers **N (1 ≤ N ≤ 100000)** & **M (N-1 ≤ M ≤ 300000)** where **N** represents the number of new cities the King has conquered and **M** represents the number of proposed roads. The cities are named with integers from **1 to N**. Each of the next **M** lines will contain three integers, **U (1 ≤ U ≤ N), V (1 ≤ V ≤ N)** & **W (1 ≤ W ≤ 1000000000)** while **U ≠ V**. Here **U** & **V** represents the name of two cities and **W** is the cost of building the bidirectional road between them.

Output

You will need to print the minimum cost to build the roads so that the King's conditions are fulfilled in the format: "**Case X: Y**" (without quotes), where **X** is the case number and **Y** is the minimum. If the proposed roads cannot fulfill the King's conditions, then simply print **"Case X: Impossible"** (without quotes) where **X** is the case number.

Sample Input	Output for Sample Input
3 2 1 1 2 1 3 3 1 2 1 2 3 2 3 1 1 4 3 1 2 1 2 3 1 3 4 1	Case 1: 1 Case 2: 2 Case 3: Impossible

Problem D

Nim Heaps

In the game of Nim, there are several heaps of stones. Two players make moves alternately. On each turn, the current player selects any heap and takes some positive number of stones from it. The player who takes the last stone wins the game.

Alice and Bob love playing the Nim game. They play it so often that they learned how to determine the winner at a first glance: if there are $a_1, a_2, a_3 \cdots a_n$ stones in the heaps, the first player wins if and only if the bitwise xor $a_1 \oplus a_2 \oplus \cdots \oplus a_n$ is nonzero.

They got so bored playing regular nim, that they decided to come up with a variation. Before starting the game, Alice and Bob agree upon two integers n and k. Then the game proceeds as follows -

- First Alice makes n heaps of stones. She can choose the number of stones in each heap. Each heap should have a positive number of stones not exceeding $2^k - 1$. Furthermore, the number of stones in the heaps should be in non-decreasing order. Formally, If Alice chooses $a_1, a_2, a_3 \cdots a_n$ as the number of stones in the heaps, then
$$1 \leq a_1 \leq a_2 \leq \cdots \leq a_n \leq 2^k - 1$$
- Then Bob chooses a nonempty subset of heaps and removes the others.
- Then the regular Nim game starts on the chosen heaps with Alice to move first.

Alice wants to know how many ways she can choose the number of stones in the heaps such that no matter how Bob responds, she will always win. Since this number can be large she is happy to know its value **modulo 998244353.**

Input

The first line will contain a single integer T $(1 \leq T \leq 10^5)$ denoting the number of test cases.
Each test case consists of one line containing two integers n and k $(1 \leq n, k \leq 10^6)$.

Output

For each test case, print a single integer in a new line - the number of ways Alice can choose the heaps **modulo 998244353.**

Sample Input	Output for Sample Input
3 1 3 2 2 3 2	7 3 0

Explanation

In the first case, there is only one heap and Alice can choose anywhere between 1 to 7 stones for that heap.

In the second case, there are 2 heaps and each heap can have at most 3 stones. The two heaps cannot have the same number of stones, otherwise Bob can choose the entire set and win. Considering this, there are 3 ways to assign the number of stones in the heaps - (1, 2), (1, 3), (2, 3).

In the third case, there are 3 heaps and each heap can have at most 3 stones. No two heaps can have the same number of stones, otherwise Bob can pick those two heaps only and win. Therefore, the number of stones must be 1, 2, 3. But then the entire set of heaps has a xor-sum of zero and Bob can pick the entire set. Thus Alice has no way of assigning the heaps so that she wins.

Problem E

Min Cost Sort

You are given a permutation $\mathbf{a(a_1....a_n)}$. You want to sort it. You can make 0 or more operations, in one operation you can choose any two indices **i** and **j** such that $\mathbf{1 \le i < j \le n}$ and swap $\mathbf{a_i}$ and $\mathbf{a_j}$ with cost **i+j**. The total cost to sort the permutation is the sum of the cost of all the operations you make. You have to output the minimum possible cost to sort the permutation.

A permutation of size **n** is a sequence of **n** integers such that every integer from **1** to **n** exists exactly once in the sequence.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 5)** denoting the number of test cases. Each test case will start with a line having a single integer **N (1 ≤ N ≤ 300)**, denoting the size of the permutation. The following line will contain **N** integers denoting the permutation.

Output

For each test case, print a single integer, the minimum cost to sort the permutation.

Sample Input	Output for Sample Input
4 3 1 2 3 3 3 2 1 3 2 3 1 3 1 3 2	0 4 7 5

Explanation:

Explanation for 3rd case: Initial permutation is,

2 3 1

We make one swap between 1st and 2nd index with cost (1+2). Now the permutation is,

3 2 1

We again make a swap between, 1st and 3rd index with cost (1+3) Now the permutation is,

1 2 3

Total Cost is (1+2) + (1+3) = 7

Problem F

Power Sequence

You are given an integer **N** and an array of **N** integers, $B_1, B_2, B_3, \ldots B_N$.

A sequence of integers $X = [X_1, X_2, X_3, \ldots X_k]$ is considered valid if

- Every integer is between 1 and N. Formally, $1 \le X_i \le N$ for all i.
- Their sum is equal to **N**. Formally, $X_1 + X_2 + X_3 + \ldots + X_k = N$.

For a valid sequence $X = [X_1, X_2, X_3, \ldots X_k]$, the beauty of the sequence is calculated as

$$F(X) = B_{X_1} + B_{X_1}B_{X_2} + B_{X_1}B_{X_2}B_{X_3} + B_{X_1}B_{X_2}B_{X_3}B_{X_4} + \ldots + B_{X_1}B_{X_2}B_{X_3}..B_{X_k}$$

More formally, $$F(X) = \sum_{i=1}^{k} \prod_{j=1}^{i} B_{X_j}$$

You need to find the sum of the **M-th** power of the function **F** for all possible valid sequences. More formally you need to find,

$$\sum_{S \in \{all\ possible\ valid\ sequences\}} F(S)^m$$

As the answer can be very large print it modulo **998244353**.

Input

First line will contain an integer **T(1 ≤ T ≤ 20)** which represents the number of test cases. For each test case there will be two lines of inputs. First line will contain two space separated integers **N(1 ≤ N ≤ 32000)** and **M(0 ≤ M ≤ 20)**. Second line will contain **N** space separated integers where the i-th integer will represent B_i ($1 \le B_i \le 10^5$).

You may safely assume that, sum of N for all over test cases does not exceed 32000.

Output

For each case, print one line with **"Case <x>: <y>"**, where **x** is the case number and **y** is the answer modulo **998244353**. Please see the sample output for more details.

Sample Input	Output for Sample Input
2 3 1 2 4 1 3 2 2 4 1	Case 1: 37 Case 2: 441

Explanation:

here for the second test case, N = 3 and M = 2.

You can make 4 valid sequences. They are [3], [1, 2], [2, 1], [1, 1, 1].

- $F([3])^2 = (B_3)^m = 1^2 = 1$
- $F([1, 2])^2 = (B_1 + B_1 * B_2)^m = (2 + 2 * 4)^2 = 10^2 = 100$
- $F([2, 1])^2 = (B_2 + B_2 * B_1)^m = (4 + 4 * 2)^2 = 12^2 = 144$
- $F([1, 1, 1])^2 = (B_1 + B_1 * B_1 + B_1 * B_1 * B_1)^m = (2 + 2 * 2 + 2 * 2 * 2)^2 = 14^2 = 196$

answer = 1 + 100 + 144 + 196 = 441

Problem G

Creative Kindergarten

Sharif, a 4 years old kid who lived in Dumbland, was very eager to know new things and always kept asking a lot of questions. One day his parents decided to admit him to the best school in the county. So, they admitted Sharif to a school named “Creative Kindergarten”, the best institute for the kids in Dumbland.

Sharif was really excited after getting admission. In his very first class, a math teacher came into the class and wrote “**2, 3, 5, 7, 11, 13, 17 and 19 are prime numbers in between 1 to 20**”.

Sharif asked - “What is a prime number Sir?” Teacher replied-” You asked a question! Just memorize it.”

In the next class, the teacher asked Sharif to list out **any K prime** numbers between 1 to 20. In this problem, you need to verify Sharif’s answer. Sharif’s answer will be considered correct if he can list out **K distinct prime numbers between 1 to 20 in any order**.

Input

The first line will contain **T (1 ≤ T ≤ 1000)**, the number of test cases. The first line of each test case will contain an integer **K (1 ≤ K ≤ 8)**. The next line contains **K** space-separated distinct integers, denoting Sharif’s answer.

Output

For each case, print one line starting with “**Case <x>:** ”, where **x** is the case number. If Sharif's answer is correct, then print “**Yes**”, Otherwise Print “**No**”.

Sample Input	Output for Sample Input
3 4 2 3 5 7 5 3 5 2 17 11 5 3 5 2 7 4	Case 1: Yes Case 2: Yes Case 3: No

Problem H

Is this Graph?

We know you all love short problem statements, so here's a treat for you.

You are given an undirected, unweighted graph of N nodes with M edges. Then you have to process Q operations of two types, each of which is an update or a query. They will follow the format described below.

a. Update: **1 U V**
For each update, you will add a new edge connecting nodes U and V.

b. Query: **2 U V**
For each query, you need to tell how many edges are there in the graph such that, if you remove any of those, then U and V will be disconnected. If **U** and **V** are already disconnected, output -1.

Input

Input starts with an integer **T,** the number of test cases. The first line of each test case starts with two integers **N** and **M**, indicating the number of nodes and edges of the graph respectively. Then each of the next **M** lines will contain two integers **U** and **V**, meaning there's an edge between these two nodes. In the next line, there will be one Integer **Q**, the number of operations. Each of the next **Q** lines will describe either an update or query which will follow the format described above. There won't be any duplicate edges.

Constraint

- **1 ≤ T ≤ 100**
- **2 ≤ N ≤ 100000**
- **1 ≤ M ≤** $min(100000, \frac{N \times (N-1)}{2})$
- **1 ≤ U, V ≤ N, U ≠ V**
- **1 ≤ Q ≤ 100000**
- **The summation of all N, M, and Q in a test file will be less than or equal to 300000.**

Output

Print the case number in a single line. Then for every query, you need to print the answer to that query in each line.

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

Problem I

Beautiful Blocks (Easy)

The only differences between the easy and the hard version are the board size (N), the number of boards (K), and the sum of N x K over all test cases.

Alice likes playing a game on an **N x N (N is even)** board where each cell contains some points. Rows are numbered from **1** to **N** from the top to the bottom, and columns are numbered from **1** to **N** from the left to the right. She starts from the **(1, 1)** cell and ends her journey on the **(N, N)** cell. But on each move, she can only move in the right or down cell. When she is on a cell, she collects all the points in that cell. Alice loves points a lot. So she always moves in such a way that the total number of points she can collect is maximized.

Bob, the magician, has **K empty N x N** board. He also has $\frac{KN^2}{4}$ small **(2 x 2)** blocks where each cell of the blocks contains some points. He can magically shuffle the cells of any block. After the shuffling, he fills the empty boards with blocks in any way he wishes and gives the filled boards to Alice. **Alice then plays her game on all of the boards.** As Bob is a kind magician, he wants to make Alice happy as much as possible. So, he shuffles the cells of the blocks and fills the empty boards with the blocks in such a way that Alice can get the maximum number of points. Can you find out how many points Alice will get?

Input

The first line will contain a single integer **T (1 ≤ T ≤ 300)** denoting the number of test cases. For each test case, the first line will have two space-separated integers **N (2 ≤ N ≤ 300 and N is even)** and **K (K = 1)**, denoting the dimension of the empty boards and the number of boards respectively. The description of $\frac{KN^2}{4}$ small **(2 x 2)** blocks follows in the next $\frac{KN^2}{2}$ lines. Each block is represented in two lines, where each line contains two integers denoting the points (**0 ≤ points in each cell ≤ 10^9**) in the cells of this block. **The sum of N * K over all test cases does not exceed 600.**

Output

For each test case, print the maximum number of points Alice will get in a single line. Please see the sample for details.

Sample Input	Output for Sample Input
2 2 1 1 4 2 3 4 1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	9 86

Explanation:

For the first case:
There is only one block,

1	4
2	3

After Bob arranges points in the block and fills the empty board, the board looks like this:

4	1
3	**2**

The cells in bold represent the path followed by Alice.
So the maximum number of points obtained by Alice = 4 + 3 + 2 = 9.

For the second case:

There are four blocks,

1	2
3	4

5	6
7	8

9	10
11	12

13	14
15	16

After shuffling the cells in the blocks, the four blocks look like this:

1	2
3	4

6	5
8	7

12	10
9	11

16	13
15	14

Then Bob fills the empty board as the following,

16	13	6	5
15	**14**	**8**	7
1	2	**12**	**10**
3	4	9	**11**

The cells in bold represent the path followed by Alice.
So the maximum number of points obtained by Alice = 16 + 15 + 14 + 8 + 12 + 10 + 11 = 86.

Problem J

Beautiful Blocks (Hard)

The only differences between the easy and the hard version are the board size (N), the number of boards (K), and the sum of N x K over all test cases.

Alice likes playing a game on an **N x N (N is even)** board where each cell contains some points. Rows are numbered from **1** to **N** from the top to the bottom, and columns are numbered from **1** to **N** from the left to the right. She starts from the **(1, 1)** cell and ends her journey on the **(N, N)** cell. But on each move, she can only move in the right or down cell. When she is on a cell, she collects all the points in that cell. Alice loves points a lot. So she always moves in such a way that the total number of points she can collect is maximized.

Bob, the magician, has **K empty N x N** board. He also has $\frac{KN^2}{4}$ small **(2 x 2)** blocks where each cell of the blocks contains some points. He can magically shuffle the cells of any block. After the shuffling, he fills the empty boards with blocks in any way he wishes and gives the filled boards to Alice. **Alice then plays her game on all of the boards.** As Bob is a kind magician, he wants to make Alice happy as much as possible. So, he shuffles the cells of the blocks and fills the empty boards with the blocks in such a way that Alice can get the maximum number of points. Can you find out how many points Alice will get?

Input

The first line will contain a single integer **T (1 ≤ T ≤ 1000)** denoting the number of test cases. For each test case, the first line will have two space-separated integers **N (2 ≤ N ≤ 1000 and N is even)** and **K (1 ≤ K ≤ 1000)**, denoting the dimension of the empty boards and the number of boards respectively. The description of $\frac{KN^2}{4}$ small **(2 x 2)** blocks follows in the next $\frac{KN^2}{2}$ lines. Each block is represented in two lines, where each line contains two integers denoting the points (**0 ≤ points in each cell ≤ 10^9**) in the cells of this block. **The sum of N * K over all test cases does not exceed 2000.**

Output

For each test case, print the maximum number of points Alice will get in a single line. Please see the sample for details.

Sample Input	Output for Sample Input
3 2 1 1 4 2 3 4 1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 2 2 1 2 3 4 1 2 3 4	9 86 18

Explanation:

For the first case:

There is only one block,

1	4
2	3

After Bob arranges points in the block and fills the empty board, the board looks like this:

4	1
3	**2**

The cells in bold represent the path followed by Alice.

So the maximum number of points obtained by Alice = 4 + 3 + 2 = 9.

For the second case:

There are four blocks,

1	2
3	4

5	6
7	8

9	10
11	12

13	14
15	16

After shuffling the cells in the blocks, the four blocks look like this:

1	2
3	4

6	5
8	7

12	10
9	11

16	13
15	14

Then Bob fills the empty board as the following,

16	13	6	5
15	**14**	**8**	7
1	2	**12**	**10**
3	4	9	**11**

The cells in bold represent the path followed by Alice.

So the maximum number of points obtained by Alice = 16 + 15 + 14 + 8 + 12 + 10 + 11 = 86.

For the third case:

The filled boards look like this:

2	1
3	**4**

2	1
3	**4**

So the maximum number of points obtained by Alice = 2 + 3 + 4 + 2 + 3 + 4 = 18.

Problem A (K^{th} Permutation Revisited)

Setter: Mohammad Ashraful Islam
Tester: Muhiminul Islam Osim,
Alter: Muhammad Ridowan, S.M. Shaheen Sha
Category: Ad Hoc

Editorial: You have find the summation of last M integers (numbers) of K^{th} permutation of consists of N integers numbered from 1, 2, 3, … , N. But, K $\leq$10 .
As, K $\leq$10, for any sequence, 1, 2, 3, 4, 5, …, N-4, N-3, N-2, N-1, N all the permutation/changes will be observed in last 4 integers (at most).
So, you can always take last $min(4, N)$ numbers and perform next permutation until you reach K^{th} permutation. Rest of the part can be done using a simple math formula to calculate the summation of series.

Problem B (Yet Another Suffix Array Problem)

Setter: Mohammad Solaiman
Tester: Raihat Zaman Neloy, Muhiminul Islam Osim
Alter: Md Mahamudur Rahaman Sajib
Category: Binary Search

Editorial: When we sort the subarrays lexicographically first the subarrays that start with 1 will come, then 2, then 3 and so on. When two subarrays have the same left endpoint, the one with smaller length is also lexicographically smaller than the other. For each value we know the number of subarrays that will have this value's position as the left endpoint. Calculate cumulative sum of these values. Now, we can do binary search on this array to find which value will be in the left endpoint of our desired subarray. After that, we can easily find the right endpoint too.

Problem C (Connect the Cities)

Setter: Raihat Zaman Neloy
Tester: Md. Imran Bin Azad, Muhiminul Islam Osim
Alter: Aminul Haq
Category: Greedy Observation

Editorial: If we observe the structure of the tree that satisfies the two conditions, we will find it to be a star. So, we just need to find a node which has N-1 nodes adjacent to it, now just take the sum of N-1 edges and find the minimum cost.

Problem D (Nim Heaps)

Setter: Pritom Kundu
Tester: Anik Sarker
Alter: Rafid Bin Mostofa
Category: Math

First let us solve the problem ignoring the condition that the array must be non decreasing. Suppose we have fixed the first i elements, let us count the number of ways we can choose the (i+1)th element. There are 2^i subsets consisting of the first elements. No two of them can have the same xor sum. The (i+1)th element cannot be one of these 2^i values. Thus there are $2^k - 2^i$ possible values for the (i+1)th element.

Now note that, no two elements can be equal (otherwise, those two have xor sum of 0). Thus every correct sequence will be strictly increasing and hence distinct. Thus we can simply divide our previous answer by $n!$. Thus our answer is

$$f(n, k) = (2^n - 1)(2^n - 2) \cdots (2^n - 2^{k-1})$$

This is an O(k) solution. To optimise it, let us rewrite it as follows.

$$f(n, k) = 2^{k(k-1)/2}(2^n - 1)(2^{n-1} - 1) \cdots (2^{n-k+1} - 1)$$
$$= F(k) \frac{G(n)}{G(n-k)}$$

Where, $F(k) = 2^{k(k-1)/2}$ and $G(k) = \prod_{i=1}^{k}(2^i - 1)$

Both F and G can be precalculated in O(k) time beforehand. Then queries can be answered in O(1).

Problem E (Min Cost Sort)

Setter: Arghya Pal
Tester: Nafis Sadique
Alter: Hasnain Heickal, Muhiminul Islam Osim
Category: Greedy, Graph, Math

The key to solve this problem is to think in terms of the permutation cycle. For the cycle containing 1, it's always better to swap the element with 1. For other cycles, either merge it with 1's cycle initially or make all the swaps with the smallest element in the cycle. Do which one gives you the smaller cost.
https://ideone.com/XRBZim

Proof: We'll think about everything in terms of the permutation cycle. **Any swap either breaks one permutation cycle into two or merges two permutation cycles into one.**

So we are given some cycles, our goal is to do some operations to reduce each permutation cycle to size 1.

Let's think what is the optimal way to break a cycle of length k into k cycles of length 1(ignore everything outside this cycle for now). Assume the elements are a_1, a_2….a_k. We need at least k-1 operations. The ultimate sum would be result of at least 2*(k-1) numbers and each of a_1, a_2…..a_k would exist at least once in the sum. And the remaining k-2 numbers would be no smaller than min(a_1, a_2…..a_k). So the sum can not be smaller than

$$a_1+a_2+....a_k + (k-2)*\min(a_1, a_2.....a_k)$$

and this is achievable. This is important and the whole problem is kind of dependent on this idea.

Assume there are k cycle's initially.
s_i = set of all elements initially in the i'th initial cycle
m_i = min of all elements in s_i
$|s_i|$ = size of set s_i
sum(s_i) = sum of all elements of in set s_i

We use set s_i to denote the i'th initial cycle to avoid confusion as cycles are merging and breaking.

Consider an optimal sequence of sorting. In each operation we are breaking a cycle or merging two cycles. Consider a graph G with k nodes where each node represents a set and an edge between node u and v represents that after some operation during the sort process, some element from s_u and some element from s_v got into the same cycle.

This graph has some connected components, different connected components don't interfere with each other, so we can think of them independently.
Let's say there is a connected component with c nodes and we try to calculate the lowest cost for sorting elements within this component's set. Assume the sets associated with nodes in this component are s_1, s_2…s_c.

As there are c nodes, then there must have been at least c merge operations and as at least one node from each set has participated in the merge operation,
minimum cost for merge operations would be >= $m_1+m_2+...m_c + (c-2)*\min(m_1,m_2,...m_c)$

There would be at least $|s_1|+...+|s_c|$ - 1 break operation and again all the elements in all the sets must participate in the break operation at least once, so minimum cost for breaking would be >= $sum(s_1)+...+sum(s_c) + (|s_1|+...+|s_c| - 1)*min(m_1,...m_c)$.

And both of these minimums are achievable.

Let's call M = $min(m_1,...m_c)$ and S is the set containing M.

What we can do is that we merge all the other sets into C. Then just start breaking this big cycle, every operation would be between M and some other element x, resulting in separating x from the cycle.

So the big picture turns out to be that we can achieve the lowest cost by merging all the sets in this specific connected component(except S) into the set containing the smallest element(which is S here). And then separating out each element one by one. So from the standpoint of a specific set s_i(except S), we can view it like this$_i$ at first we are merging it into S with cost $M+m_i$ and then making $|s_i|$ operations with total cost $|s_i|*M + sum(s_i)$, occuring a total cost of,

$$M*(|s_i|+1) + sum(s_i) + m_i$$

And for S we are getting cost

$$sum(S) + (|S|-2)*M.$$

This point of view enables us to get rid of mathematical equations from the rest of the proof.

So what we have established till now is that, there exists an optimal sequence of operation where we can group the permutation cycles initially, then for each group we'll just merge all cycles into the cycle containing the smallest number and then break that big cycle one element at a time. And different groups don't interfere with each other. Also cost for these operations can be thought as independent per cycle. Now it is easy to see that for any cycle it is better to merge it to 1's cycle than any other cycle. So for each group we can just break the cycle containing the smallest number within itself and merge the rest of them to 1's cycle and break it later. So how to check weather for a cycle s_i if we should merge it to 1's cycle or break it within itself?

Breaking it within itself gives us cost $|s_i-2|*m_i + sum(s_i)$

And merging it into 1's cycle and later breaking it gives us cost, $1+m_i + |s_i| + sum(s_i)$

Just do whichever gives you the smaller cost.

Problem F (Power Sequence)

Setter: Md Mahamudur Rahaman Sajib

Tester: Anik Sarker, Pritom Kundu

Alter: Rafid Bin Mostafa
Tags:Dynamic Programming, Math, Online FFT

here beatiness *F* function can be written as,

$F(\{x1,\ x2,\ x3 \ldots . xk\}) = B[x1] * (1 + B[x2] * (1 + B[x3] * \ldots..))]$

Let's try to solve this problem for m = 1.

$dp[n] = sum\ of\ beautiness\ function\ F\ for\ all\ possible\ such\ that\ sum\ is\ n$

$$dp[n] = \sum_{i=1}^{n} B[i] * (1 + dp[n - i])$$

Here time complexity of this naive dp solution is $O(n^2)$.

But if we think of 3 polynomials,

$$G(x) = \sum_{i=0}^{n} B[i] * x^i$$

$$H(x) = \sum_{i=0}^{n} (1 + dp[i]) * x^i$$

$$Q(x) = \sum_{i=0}^{n} dp[i] * x^i$$

then we can easily see,

$Q(x) = G(x) * H(x)$

But the problem is G(x) is static but H(x) is dynamic and H(x) itself depends on the Q(x) so polynomial multiplication using FFT is not possible here. To solve this problem faster way we need to use the trick of online FFT(where we can trickily do polynomial multiplication block by block where block size will be power of 2)(Online FFT), then time complexity will be reduced to $O(nlog(n)^2)$.

But, how can we solve the problem for any m? First, the observation is that m is not too big. Let's try to sketch the dp formula.

$dp[m][n] = sum(F(seq)^m)\ for\ all\ possible\ sequence\ such\ that\ sum\ is\ n$

$$dp[m][n] = \sum_{i=1}^{n} B[i]^m * \sum_{j=0}^{m} mCj * dp[j][n - i]$$

Time complexity of this dp solution is $O(n^2 * m^2)$ which is too slow.

Let's rewrite this equation for m = p,

$$dp[p][n] = \sum_{i=1}^{n} B[i]^p * \sum_{j=0}^{p} pCj * dp[j][n - i]$$

For different p, we can derive polynomials

$$G_p(x) = \sum_{i=0}^{n} B[i]^p * x^i$$

$$H_p(x) = \sum_{i=0}^{n} (\sum_{j=0}^{p} pCj * dp[j][i]) * x^i$$

$Q_p(x) = G_p(x) * H_p(x)$

so, here we need to do online fft for different p which is a bit tricky to code. But there is a small problem to build $H_p(x)$. If we iterate over j every time we build $H_p(x)$ so time complexity of total builds of $H_p(x)$ is $O(nm^2 log(n))$ and online FFT part's time complexity will be $O(nmlog(n)^2)$. So time complexity is $O(nm^2 log(n) + nmlog(n)^2)$.

Problem G (Creative Kindergarten)

Setter: Mohammad Ashraful Islam
Tester: S.M. Shaheen Sha
Alter: Mohammad Ridowan
Category: Adhoc

Problem H (Is this Graph?)

Setter: Aminul haq
Tester: Arghya Pal
Alter: S.M. Shaheen Sha, Md. Mahamudur Rahaman Sajib, Raihat Zaman Neloy
Tags: Data Structure, DSU+Link Cut Tree or DSU+HLD+Segment Tree/BIT
Solution: There are several solutions to solve this problem, we will discuss the offline approach using DSU and HLD.
First, let's understand the problem. It simply asks to count the number of bridges between two nodes in each query. But the challenge is the updates. When we are adding a new edge between two nodes, and if those two nodes are already connected, then after adding it, there won't be any bridges between them, as it will become a biconnected component.
So, to solve it, first, we will read all the full input and make the tree (or forest). How to do it? Using DSU, we will add an edge to our tree if those two nodes are not connected already. Now we have a tree (or several trees which make a forest). We can build an HLD structure from these trees. Now we will iterate over the entire input again. When we get a tree-edge (from previous DSU) we will set this path's value to one. And when we get a non-tree-edge, we need to set zero to all the edges in that path, as this new edge will remove all the bridges between these current two nodes. And for a query will count how many 1's are there in that path (or simply, the total sum of values in that path).

https://ideone.com/6CbfDh
https://ideone.com/3ACBYU

Problem I (Beautiful Blocks (Easy))

Setter: Ashiqul Islam
Tester: Arghya Pal
Alter: Pritom Kundu
Tags: Observation, DP
Solution:

Let's say,
A block is X-block if the path has X cells in the block.

The path goes through exactly (N-1) blocks.
Among them:

(1) K blocks are 3-block
(2) (K-1) blocks are 1-block
(3) the rest are 2-block

So, we can dp on,
(block_index, how many blocks has been taken, how many 3-blocks have been taken, how many 1-block has been taken)

the answer would be f(no_of_blocks, N-1, K, K-1) for all K.

this is too slow. instead of keeping both the count of 3-blocks and 1-blocks, we can keep only their difference.

(block_index, how many blocks has been taken, difference between 3-block count and 1-block count)

This is a n^4 dp. This is also too slow.

Instead of considering all the blocks in the dp, we can only consider O(N) blocks. The relevant blocks can be found as follows:

Let's say for block, a1 = its max, a2 = 2nd max, a3 = 3rd max, a4 = 4th max.

Let's sort the blocks three times in decreasing order of: a1, (a1+a2), (a1+a2+a3).

The relevant blocks are the union of the first (n-1) blocks from all the 3 sorted arrays.

Overall Complexity (n^3)

Interestingly, there is another solution. You can notice that in an optimal selection, the number of 3-blocks and 1-blocks are almost equal. So if you random shuffle all the blocks, then the max difference of 3-blocks and 1-blocks in an optimal assignment over all the prefix won't be very big. So you can reduce that difference dimension to something like 20.

Github Link:
https://github.com/Contest-Problems/beautiful_blocks

Problem J (Beautiful Blocks (Hard))

Setter: Arghya Pal & Ashiqul Islam
Tester: Arghya Pal
Alter: Pritom Kundu
Tags: Observation, DP
Solution:

Now the solution for easy version is too slow for the harder version. Let's solve for K = 1. You can easily make it work for the given Ks.

Let's sort the blocks in descending order of sum of their best+2nd best block. Now we can observe that for any two indices i,j such that (1<= i < j <= n), it's always better to take the i'th block as 2-block rather than the j'th block. So there exists a prefix of the blocks where we will take all the blocks and from the rest we will only take 1-blocks and 3-blocks.

So we need to calculate two things for each prefix,
i) if we take all the blocks in this prefix, for each different value of (3_blocks - 1_blocks), what is the maximum we can achieve? We can do it in O(n^2) complexity.

ii) what is the maximum we can achieve if we take a fixed number(N-1 - length of the prefix) of blocks from the rest of the blocks and again we need to know it for all different values of (3_blocks - 1_blocks). It turns out that for a suffix we can calculate this value in O(nlogn) complexity. So for n suffixes, total complexity sums up to be O(n^2*logn).

Let's see how we can do the 2nd part.
Now assume we are given **m** blocks, we have to take exactly k block from them and we need to know, what is the maximum we can achieve from

(1) k 1-blocks and 0 3-blocks
(2) k-1 1-blocks and 1 3-blocks
.....
.....
.....
(k+1) 0 1-blocks and k 3-blocks

If we have to take k 1-blocks and 0 3-blocks, it's obvious we will take the k blocks which have the highest a1(a1 denotes the maximum cell in the block).

Now if we need to take k-1 1-blocks and 1 3-blocks, we can either turn one 1-block into 3-block or remove a 1-block and take a previously skipped block as 3-block.

In general if we have a set of 1-blocks and 3-blocks which gives us best result for (x,k-x) configuration(x 3-blocks and k-x 1-blocks), then for obtaining best result for (x+1,k-x-1) configuration we can just turn one 1-block into 3-block or remove a 1-block and take a previously skipped block as 3-block. This idea seems very intuitive but the life of a problem-setter is not so easy, you have to prove it also :(

(Un)fortunately the proof is ~~easy~~ hard in this case.(changed my mind while writing the proof, not easy to write, definitely not easy to read)

Essentially, what we are doing is that while going from (x,k-x) configuration to (x+1,k-x-1) configuration, we are keeping the set of 3-blocks intact and just adding another 3-block, also just changing exactly one 1-block(either turning it into 3-block or removing it).

Lets assume,

A is the set of skipped blocks, B is the set of 1-blocks, C is the set of 3-blocks for some (x, k-x) configuration. .....(assumption 1)

One thing that we can observe is that going from (x,k-x) to (x+1,k-x-1) can be done without any “direct exchange” between sets.

Direct exchange between two sets X, Y takes place if at least one block from X goes to Y and one block from Y goes to X.

Let’s say A_1, B_1 and C_1 corresponding sets for 0-blocks, 1-blocks and 3-blocks which gives us maximum result for (x+1, k-x-1) configuration and among all such sets C_1 is a set such that $|C_1$ intersection $C|$ is maximum. If C is not a subset of C_1, that means there was at least one block c in C that is now at some other set.

i) If c is now in B_1, then
   1) Either there exist a block b in B which is now at C_1, or
   2) There exist a block b in B which is now at A_1 and there is a block a in A which is now at C_1

ii) If c is now in A_1, then
   1) Either there exist a block a in A which is now at C_1, or
   2) There exists a block a in A which is now at B_1 and there is a block b in B which is now at C_1.

In all these cases we can cyclically reverse the position of the blocks and we will achieve a better or equal result.(Otherwise assumption 1 can not be true)

We can do this as long as C is not a subset of C_1, every time we will be increasing the $|C$ intersection $C_1|$.

In the same way, we can see that if any block goes from B to A_1 and a block goes from A to B_1, we can reverse this as well.

Github Link:
https://github.com/Contest-Problems/beautiful_blocks

MIST-National Collegiate Programming Contest, 2020

22nd February 2020

You get 16 Pages, 11 Problems & 300 Minutes

Sponsored By:

Problem A

Find the Word

You are given a string **S** of lowercase English letters. Let, **X** and **Y** be two substrings of S such that:

$$\mathbf{X} = S_A S_{A+1} \ldots\ldots S_{B-1} S_B \text{ and } \mathbf{Y} = S_C S_{C+1} \ldots\ldots S_{D-1} S_D$$

Given the values of **A**, **B**, **C** and **D** as input, you have to answer how many times **Y** occurs as a **substring** of **X**.

Input

The first line of the input contains an integer **T** (**1 ≤ T ≤ 5**) denoting the number of test cases.
Each test case is described as follows :

- The first line contains **S (1 ≤ |S| ≤ 10^5)** - the given string.
- The second line contains **Q (1 ≤ Q ≤ 10^5)** - the number of queries for this string.
- Next **Q** lines each describes a query **A B C D (1 ≤ A ≤ B ≤ |S|, 1 ≤ C ≤ D ≤ |S|)**

Output

For each query, output the number of times **Y** occurs as a **substring** in **X**.

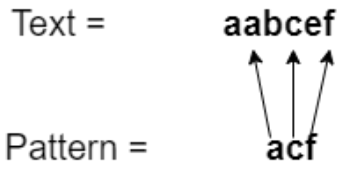
Sample Input	Output for Sample Input
3 ababab 3 2 4 4 5 3 6 1 4 5 6 2 3 abcac 2 1 3 4 5 1 3 5 5 aaaaa 1 1 5 1 2	1 1 0 0 1 4

Problem B

Search The Files

Kaishya was fed up with all the IDEs he uses. So, he wanted to build a great IDE which he calls **AIDE (Artificially Intelligent Development Environment)**. He started coding to build this legendary **AIDE**. But at one stage, he stopped. He was trying to implement a feature to search files from the root of the project folder. But he was not sure which algorithm to implement for that. Kaishya is familiar with a lot of string matching algorithms, but the disadvantage of those is - he will need an exact keyword to find out a file. So what he needs is an algorithm that will do the approximate matching.

After Googling for a few hours, Kaishya came to know about a new term called "**Fuzzy Search**". This algorithm tries to find if the search **pattern** is a subsequence of the main **text**. For example, **acf** is a subsequence of **aabcef** but **acb** or **aba** are not.



Though Kaishya learned this algorithm, he doesn't have much time to implement this as he will invest his precious time in developing other features of AIDE. From somewhere he got information about you guys. So, he is asking for your help. You will be given **N** file paths and a search **pattern**. A file path is a string representing where the associated file is located. Each file path is assigned an unique id between **1** to **N**. You need to find all the file paths which will contain the **pattern** according to the fuzzy search algorithm.

Input

The first line of the input will contain a single integer **T (1 ≤ T ≤ 5)** which denotes the number of test cases. Each test case starts with a line containing a single integer **N (1 ≤ N ≤ 10^5)**. Each of the next **N** lines will contain one **file path (1 ≤ |File path| ≤ 10^5)** as a string. Id of the i^{th} file path is i. After that, there will be one line that contains the **pattern (1 ≤ |pattern| ≤ 10^5)**. The file paths and the pattern will only consist of lowercase letters and one special character "*/*" (ASCII 47). You should assume that the sum of length of all the strings over all test cases in a single input file will never cross the limit **10^6**.

Output

For each test case, the first line of the output should be the case number in the format "**Case X:**" (without quotes) where **X** is the case number. The next line of the output should contain the ids of the files in increasing order, separated by a single space that contains the pattern according to the fuzzy search algorithm. Print "**No files found!**" (without quotes) if you can't find any such file path.

Sample Input	Output for Sample Input
1 3 /root/documents/a /root/downloads/b /tmp/downloadedfile rs	Case 1: 1 2

Problem C

Bracket Colouring

Do you know what Araspa loves more than a bracket sequence? **A Balanced Bracket Sequence**. Today she is constructing them. She will make a balanced Bracket sequence of **2 * N** length. To make it more interesting she will use different types of brackets. She has an **infinite** number of brackets of **infinitely many** types.

A balanced bracket sequence is a string consisting of only brackets defined formally below:

- The empty string is a balanced bracket sequence.
- if **S** is a balanced bracket sequence, then so is "Opening Bracket of any **i^{th}** type" + **S** + "Closing bracket of **i^{th}** type".
- if **S** and **T** are balanced bracket sequences, then so is **S + T**.

Here, **'+'** means concatenation of string.

Now she is asking you to find the distinct number of balanced bracket sequences of **2 * N** length you can make**.** It should be infinity right?! How can you calculate infinity**?**

To make your life easier she will only permit you to use **X** types of brackets of her own choosing and you will have to use all of them **at least once** in every distinct sequence you make i.e if she allows you to use only these 3 types of brackets: **()**, **{}** and **[]**; then you can make "{}([])", but you can't make "({()})" because you didn't use 3rd type of bracket here.

But to make your life a little bit easy-medium she will not ask you to do this only for a single **X.** Rather she will give you **K** and ask you to find ***DBS()*** value for all the integers from **1 to K**. The definition of ***DBS()*** function is given below:

DBS(X) = Number of distinct balanced bracket sequences you can make using all of the **X** types of brackets at least once**.** As the number of ways can be very large so we will store them using modulo **998244353**.

Now Araspa knows that printing an array is no joke so you will have to print **Func()** which is defined below:

```
Func() :
      Ans = 0
      For(X = 1 to K)
            Ans = Ans ^ DBS(X)
      Return Ans
```

Here **'^'** means Bitwise XOR operation

See the Explanation for more clarification.

Input

In the first line, there will be a single integer **T (1 ≤ T ≤ 20)**, denoting the number of test cases. Each of the next **T** lines will contain two space separated integers: **N** and **K (1 ≤ K ≤ N ≤ 10^5)**

Output

For each test case, print the case number and then print a single integer answer to that test case as described in the statement. Please note that the final result may be greater than **998244353.**

Sample Input	Output for Sample Input
6 2 2 99190 98831 100000 10 100000 100 100000 10000 100000 100000	Case 1: 6 Case 2: 107518997 Case 3: 412801477 Case 4: 442870344 Case 5: 531237973 Case 6: 168710547

Explanation :

In the first test case**,**
For **X** = 1, you can make the following 2 types of brackets :
()()
(())

For **X** = 2, you can make the following 4 types of brackets :
({})
{()}
(){}
{}()
So answer = 2 ^ 4 = 6

Problem D

Water in Tree

You are given an undirected tree of **N** nodes, where nodes are labeled from **1** to **N**, with the root in node **1**. Each node of this tree can store exactly one unit of water. Now, you need to process **Q** queries on this tree. There can be two types of queries. Here are the definitions:

Type 1: *1 U X* means - pour **1** unit of water on node **U** for **X** times. When a node receives **1** unit of water, it will try to pour the water down to its first empty child. If every child is storing water, then the node will store the water itself. If the node itself isn't empty, the water overflows. Note that, this process is true for every node that receives any water. No other water is poured until one unit of water is finished propagating down. On the tree below, fig-1 and fig-2 reflects the effects of query 1 1 3 and then query 1 2 12. Note that the shaded nodes are storing water.

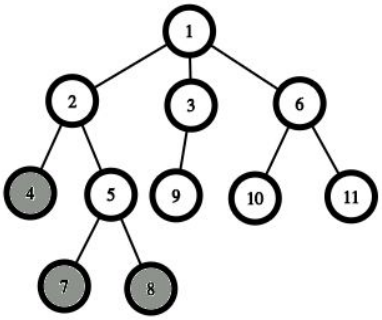


fig-1

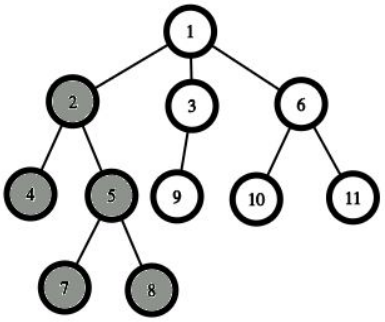


fig-2

Type 2: *2 U* means - you have to answer if node **U** has water in it or not at this particular moment.

Note that, the order of children of each node U is determined by the ascending order of their node label.

Input

The input contains **T (1 ≤ T ≤ 100)**, the number of test cases. The first line of each case contains a single integer **N (1 ≤ N ≤ 10^5)**, number of nodes in this tree. Each of the next **N - 1** lines contains two integers **U** and **V (1 ≤ U, V ≤ N)** - which means there is an edge between node **U** and node **V**. It is guaranteed that the given input is a tree. Next line contains a single integer **Q (1 ≤ Q ≤ 10^5)**, number of queries. Each line of the next **Q** lines contains either **1 U X** (if type 1) or **2 U** (if type 2), where **1 ≤ U ≤ N** and **1 ≤ X ≤ 10^8**. It is guaranteed that sum of **N** and sum of **Q** is at most **400,000** separately, in each input file.

Output

For each test case, the first line should print the test case number in the format showing in the sample output. Then for each query of type 2, you have to print **1** (if node **U** has water) or **0** (if it doesn't).

Sample Input	Output for Sample Input
1 11 1 2 1 3 1 8 2 4 2 5 5 6 5 7 3 11 8 9 8 10 8 1 1 1 1 2 2 2 5 1 1 1 2 5 1 1 4 2 3 2 10	Case 1: 0 1 1 0

Problem E

K-Divisors

The ***positive divisor function*** is defined as a function that counts the number of positive divisors of an integer **N**, including **1** and **N**.

If we define the *positive divisor function* as **D(N)**, then, for example:
D(24) = 8 (Because **24** has **8** divisors and they are **1, 2, 3, 4, 6, 8, 12, 24**)

Calculating **D(N)** is a classical problem and there are many efficient algorithms for that. But what if you are asked to find something different? Given a range and an integer **K**, can you find out for how many **N** in the given range, **D(N)** equals **K**?

Input

In the very first line, you'll have an integer called **T**. This is the number of test cases that shall follow. Every test case contains three integers, **L**, **R**, and **K**. **L** and **R** represent the range and are inclusive.

Constraints

- **$1 \leq T < 31$**
- **$1 \leq L \leq R < 2^{31}$**
- **$1 \leq K < 2^{31}$**

Output

For every test case, you must print the case number, followed by the count of numbers with exactly **K** divisors in the range.

Sample Input	Output for Sample Input
3 10 10 4 2 13 2 100 10000 100	Case 1: 1 Case 2: 6 Case 3: 0

Problem F

Jumping Frog Redefined

There was a river beside a road in Babylon. The river is divided into **M+1** blocks numbered from **0** to **M.** Some of these blocks in the river contain one stone each. At the further end of the river, there lives a frog. In order to search for food, the frog crosses the river by jumping on blocks. The frog can only perform jumps on blocks that have a stone.

The frog lives on the **0th** block, and it wants to reach the **Mth** block jumping through these blocks containing stones. Both the **0th** and **Mth** block will always contain stones. If the frog jumps from block **x** to block **y,** then he has to perform **|x - y|** unit jump. The frog gets tired if he performs a long jump. So he prefers small jumps, rather than long jumps.

Among all the **M+1** blocks, there are only **N+2** (including **0th** and **Mth** block) blocks that contain stones. And the frog memorized those block numbers. So the frog decided to put some extra stones in some of the blocks of the river to minimize his maximum jump length. Now, the frog is curious to know his lowest possible maximum jump length, if he puts **at most S** extra stones in some of the blocks of the river in **Q** different queries. Can you help him?

Input

The First line contains an integer **T(1 ≤ T ≤ 14)** denoting the number of test cases. Each test case starts with two space separated integers **N, M (1 ≤ N ≤ 10^5), (N < M ≤ 10^6)**. The next line contains **N** space separated integers A_1 A_2 A_3 ... A_N **(1 ≤ A_i < M)** denoting the blocks having a stone. All the block numbers are unique. The following line contains an integer **Q (1 ≤ Q ≤ 2 * 10^5)** denoting the number of queries. The next line contains **Q** space separated integers **S (0 ≤ S ≤ M)**, denoting the maximum number of extra stones. It is guaranteed that the sum of **N** and **Q** of over all test cases will be less than **3 * 10^6**.

Output

For each test case print the case number, and then for each query print the lowest possible maximum jump length. Please see the sample for details.

Sample Input	Output for Sample Input
1 3 30 5 10 29 3 1 2 30	Case 1: 10 7 1

Warning: Dataset is huge. Please use faster I/O methods.

Problem G

Age of Perceptrons

For research purposes, you are going to collect cancer data from the National Institute of Cancer Research & Hospital (NICRH) for **N** days. Each day they will provide you with a 2D data point that represents a tumor location in an X-Ray and its label - malignant or benign. Data will be given in the format **(X, Y, L)** where **(X, Y)** is the data point and **L** is its label. If **L = 0,** then it's malignant cancer and if **L = 1,** then it's benign. For each day, you want to figure out if the malignant and benign data points up to that day are **linearly separable**.

Two sets of 2D points are **linearly separable** if there exists a straight line so that no points are on the line and no pair of points from different sets are on the same side of the line. And also, if one of the sets is empty, then the sets are linearly separable.

Input

The first line of the input contains an integer **T** (**1 ≤ T ≤ 25**), which denotes the number of test cases. The first line of each test case contains a single integer **N** (**$1 \le N \le 10^5$**), the number of days. **N** lines will follow, i^{th} of which will contain three integers X_i, Y_i and L_i (**$-10^9 \le X_i, Y_i \le 10^9$, $0 \le L_i \le 1$**) - the data point and label for i^{th} day. It is guaranteed that all the points in a test case are distinct. It is guaranteed that the sum of **N** of over all test cases will be less than or equal **$3 * 10^5$**.

Output

For each test case, the first line should contain the case number in the format "**Case X:**" (without the quotes) where **X** is the case number. After that, for each day, print a single line containing "**YES**" or "**NO**". Print "**YES**" if the dataset up to that day is linearly separable or "**NO**" otherwise. Please see the sample for details.

Sample Input	Output for Sample Input
2 2 0 0 0 0 1 1 4 0 0 1 1 1 1 1 0 0 0 1 0	Case 1: YES YES Case 2: YES YES YES NO

Problem H

The Nostalgia of a Deja Vu

In this problem, all you have to do is generate an array, having **N non-negative** integers, which satisfies the given **Q** constraints. Every constraint is of the following kind - "The bitwise **XOR** of all the numbers from the **L**-th position to the **R**-th position of the array has to be equal to **X**". You need to find the lexicographically smallest array which satisfies all the constraints or claim that such an array does not exist.

Input

The first line of the input contains an integer **T**, denoting the number of test cases. Then the description of the **T** test cases follows. The first line of every test case will contain two integers **N** and **Q**, denoting the length of the array and the number of constraints respectively. Each of the following **Q** lines will contain a constraint of the following form: **L R X**.

Constraints

- **$1 \le T \le 15$**
- **$1 \le N, Q \le 5 * 10^4$**
- **$1 \le L \le R \le N$**
- **$1 \le X \le 2 * 10^9$**

Output

For each test case, print a line containing the case number and if such an array can be constructed then print the lexicographically smallest array which satisfies all the given constraints. Otherwise, print "**Impossible**".

Sample Input	Output for Sample Input
1 7 2 2 5 2 6 7 3	Case 1: 0 0 0 0 2 0 3

An array **C** of size **N** is called lexicographically smaller than another array **D** of equal size, if and only if there exists an **i ≤ N** such that:

C[p] == D[p] for all **1 ≤ p < i** and **C[i] < D[i]**

Warning: Dataset is huge. Please use faster I/O methods.

Problem I

Save Mozzarella!

Joker is back again! He has planned to destroy the country Mozzarella.
Mozzarella is a country of **N** cities and **N** roads. Initially, people can go to any city from any city via one or multiple roads. All roads are bi-directional. There might be multiple roads between the same pair of cities. Even both endpoints of a road can be the same.
We will say that Mozarella is **disconnected** if there exists a pair of cities, **u** and **v** such that **u is not reachable from the city v.**
To destroy Mozzarella, **Joker's ultimate goal is to make Mozzarella disconnected**. So he picks **K roads randomly** and puts bombs on those roads.

Batman, as usual, got the news. But he only knows that exactly **K roads** have been selected but doesn't know which roads got selected. And he has no time to check every road, so he also picks **K roads randomly**. For each of the **K** roads Batman has selected, he checks if this road contains any bomb. If it has any, then he disposes of the bomb.

A road will be destroyed if Joker puts a bomb in it and Batman doesn't dispose of it.

You have to find the **probability of Joker's success, S**. In other words, the probability that Mozarella got disconnected. It can be shown that **S** can be represented by **P/Q** where **P** and **Q** are co-prime integers and **$Q \neq 0 \bmod (10^9+7)$**. Find the value of **$(PQ^{-1}) \bmod (10^9+7)$**.

Input

The first line of the input contains a single integer **T ($1 \leq T \leq 200$)** denoting the number of test cases. The description of **T** test cases follows.
The first line of each test case contains two integers **N** and **K ($1 \leq K \leq N \leq 100000$)**. Each of the next **N** lines contains two space-separated integers **x ($1 \leq x \leq N$)** and **y ($1 \leq y \leq N$)** denote that cities **x** and **y** are connected by a road. Note that, the summation of all **N** in a single test file will be at most **1000000.**

Output

For each test case, print a single line containing the **probability of Joker's success, S**. For more details, please see the sample I/O.

Sample Input	Output for Sample Input
2 3 3 1 2 2 3 1 3 4 1 1 2 2 3 1 3 3 4	0 687500005

Problem J

Evil Judges

Not Competitive Programming Contest (NCPC) 2020 is going on. But this time it's going to be held in a brand new format. The format is given below:

- Scoring for a problem will be dynamic. The score for a problem will be an integer and can range from **1** to **1000000000**. The function for dynamic scoring is unknown to the contestants.
- There is no time penalty for wrong submissions.
- The time penalty will be the exact time of the latest accepted solution.
- Contestants will be sorted according to their total score, a larger score will come first. If their total scores match, then they will be sorted according to their time penalty, a smaller time penalty will come first. If they also match, they will share the position. Technically, If **X** has a larger score than **Y** or they both have the same score but **X** has a smaller time penalty, then **X** performed better than **Y**. So, the rank of a contestant is **1 + [number of contestants having better performance]**.

Rank	Name	Total Score	Time Penalty	A(3)	B(9)	C(10)	D(10)
1	ptr	23	120	120	-	23	57
2	trst	22	111	7	18	-	111
3	rng	20	107	-	-	43	107
3	bmry	20	107	-	-	89	107
3	mnvmr	20	107	-	-	107	73
6	rdsh	19	107	-	80	107	-
7	bnq	19	315	-	315	-	212

rng, **bmry**, **mnvmr** share the same rank **3** because they have the same score and time penalty thus the exact same performance. **rdsh** gets rank **6** because he has **5** contestants having better performance than his.

The contest is over. Each contestant solved some problems (probably zero, probably all) having some time penalty. The standings are not revealed yet thus no one knows the scores of the problems.

But you know the judges are evil. If a contestant bribes enough to them, they may manipulate the score for each problem in such a way that the rank of that contestant would be best possible.

In the standings above, the scoring for each problem gives the best possible position for **ptr** which is the championship. But if the judges give **11** points instead of **9** for problem **B**, then **trst** would get **24** points and will become champion in that contest. In this scenario, **rdsh** and **bnq** will also have **21** points instead of **19** and will become **3rd** and **4th** respectively. Remember, if the judges set the score for a particular problem, it's the same for all the contestants.

So, judges can set scores for the problems according to their wish. Thus different scoring distribution will generate different standings. Let's call the best possible rank for a particular contestant, **evil-rank**. As the

contestants don't know the function for dynamic scoring, judges can set the score for each problem at their wish so that the rank of the contestant is best possible. So mathematically, if there are **M** problems, then a total of $\mathbf{1000000000^M}$ different scoring distribution is possible. For a particular contestant, among all these standings, the best rank for that contestant is the evil-rank for him.

You are given the list of solved problems and the time penalty for each contestant. You have to find the evil-rank of each contestant if he bribes the evil judges individually. This means when you are calculating the evil-rank for a particular contestant, only he bribes the judges and other contestants do not.

Input

Input starts with an integer, **T**, denoting the number of test cases. For each test case, the next line contains two integers **N** and **M,** denoting the number of contestants and the number of problems. The next **N** lines contain the details for each contestant. Each line will be in this format:

[type][problems][space][penalty]

[type] will be either '**!**' or '**?**' (without the single quotations). '**!**' denotes that the contestant has solved the following problems and '**?**' denotes that the contestant has solved all the problems except the following problems.
[problems] will contain a string containing only the first **M** upper-case Latin letters. Each character of the string will be unique and this can be empty.
[space] denotes simply space character ' ' (**ASCII 32**).
[penalty] will be an integer between **0** to **1000000**.

Check the sample input for more clarification.

Output

For each test, print the case number and space-separated evil-rank of each contestant in the given order. Check the sample output for more clarification.

Constraints

1 ≤ T ≤ 50
1 ≤ N ≤ 100000
1 ≤ M ≤ 20
1 ≤ Sum of N over all test cases ≤ 1000000
1 ≤ Sum of M over all test cases ≤ 200
0 ≤ Time Penalty ≤ 1000000

Sample Input	Output for Sample Input
2 4 2 !A 49 ?A 49 !B 99 ?B 99 7 4 !ACD 120 !ABD 111 !CD 107 ?AB 107 !CD 107 !BC 107 ?AC 315	Case 1: 1 1 2 2 Case 2: 1 1 2 2 2 1 2

Problem K

Respectfully Giving Away

These days, it's very hard to get respect. People only respect a few people. Like, among the contestants, people only respect the ICPC World Finalists. This behavior is so infectious that some of the companies are giving different salaries to World Finalists. This often becomes the reason for heartache for many who took part in contests for so long but couldn't manage to get into the World Finals.

You as a contestant, want to analyze this behavior. Given a company's salary for its newly recruited **N** employees, you want to figure out if they are biased towards World Finalists. You know among the **N** newly recruited employees exactly one person **P** is an Ex-World-Finalist. And every company will give **P** the highest amount of salary. If there is at least one other newly recruited employee whose salary is equal to **P**'s salary, then the company is not biased, otherwise, it is.

Input

The first line will contain **T (1 < T < 10)**, the number of test cases. Each case will start with **N (1 < N < 101),** the number of newly recruited employees for that company. A line with **N** integers will follow. Each of these integers will be between **50** to **100** which represents the salary of each employee in thousands.

Output

For each case print one line with either **"Yes"** if the company is biased, or **"No"** if it is not.

Sample Input	Output for Sample Input
2 3 60 60 75 3 60 75 75	Yes No

RUET Inter University Programming Contest 2019

Technocracy 2019

Hosted by ECE, RUET

Rajshahi, Bangladesh

5th July 2019

You get 20 Pages, 11 Problems & 300 Minutes

Platform Support:

CodeMarshal

Problem Set By:

Bangladesh Association of Problem Setters

Problem A

Input: Standard Input
Output: Standard Output

I have short description

Given two integers **N** & **M**, output the value of $\sum_{r=1}^{M} (N \% r)$

Input

The first line of the input contains an integer **T**, denoting the number of test cases. Then the description of the **T** test cases follow. The first and the only line of every test case will contain two integers **N** and **M**.

Constraints

- **1 ≤ T ≤ 3**
- **1 ≤ N ≤ 10^{100000}**
- **1 ≤ M ≤ 10^{5}**

Output

For each test case, print a line containing the case number and the value of $\sum_{r=1}^{M} (N \% r)$.

Sample Input

```
2
193 17
2441139 505
```

Output for Sample Input

```
Case 1: 68
Case 2: 66262
```

Problem B	Input: Standard Input Output: Standard Output
Playing with capitals	

Byteland is a very versatile country. The economic condition of this country changes a lot. That's why the government of Byteland needs to change its capital very frequently. There are **N** cities and all those cities are always connected with **N-1** bidirectional roads. Each node is labelled with integers from **1** to **N**. Each node also has a value associated with it which is called **VAT** for that specific node. When someone wants to send some goods through a city **U**, then he will have to pay $\mathbf{VAT_U}$ amount of money to the government. Initially, the capital city is the node labelled with **1**.

There is a ministry of the government to handle this type of situation when economic condition changes. But to compute a few things faster, the ministry has hired you three guys to help them out with the following works:

1. Given a node **U**. You have to destroy the road connecting **U-V** where **V** is a city which is directly connected with **U** and **V** is the closest city to present capital city **C** among all the adjacent city of **U**. And then create a new bidirectional road **C-U**. After these two steps, make **U** the capital. It is guaranteed that **U** is not the present root.
2. Given two nodes **U V**. You have to find two things -
    a. Find the closest city from the capital which lies on the simple path from **U** to **V**.
    b. Figure out the total **VAT** to parcel some goods from **U** to **V**.

The government already know that you guys are great programmers. So you have to help them now.

Input:

The first line of the input contains a single line **T(1 ≤ T ≤ 3)** denoting the number of test cases.

The first line of each test case contains two integers **N(1 ≤ N ≤ 10^5)** denoting the total number of cities in Byteland and **Q(1 ≤ N ≤ 10^5)** denoting the number of tasks you have to complete. Next line contains **N** integers denoting the **VAT** of city **i(1 ≤ i≤ N & 1 ≤ VAT_i ≤ 10^5)**. Each of the next **N-1** lines contains two integers **U V (1≤U, V≤N & U ≠ V)** which means there is a direct bidirectional road between city **U** & **V**. Each of the next **Q** lines will contain any one type of tasks described above in the following formats:

1 **U**: Denoting the task type **1** where **1 ≤ U ≤ N**.

2 **U V**: Denoting the task type **2** where **1 ≤ U, V ≤ N**.

Output:

The first line of each test case should contain the case number in the format: **"Case X:"** where **X** is the case number (without quotes). For each of the task type **2**, you have to print two space-separated integers **P Q** where **P** is the closest city from the capital and **Q** is the total vat. Please see the sample input-output for more clarifications.

Sample Input

```
1
5 5
1 2 3 4 5
1 2
1 3
2 4
2 5
2 4 5
1 2
2 1 5
1 5
2 1 5
```

Output for Sample Input

```
Case 1:
2 11
2 8
5 8
```

N.B.: Input file is huge. Please use faster I/O.

Problem C

Input: Standard Input
Output: Standard Output

Binary Search

Binary search is one of the most important tools that is being used in solving problems. You are given an array **S** consists of **N** elements indexed from **1** to **N**. We want to search an element **P** within the array. Using binary search we can solve this problem very easily. Binary_search function is given as follows. This function returns **true** if P is found in the array and returns **false** if P is not found in the array.

```
function Binary_search(S, N, P):
1.    L := 1
2.    R := N
3.    while L ≤ R:
4.        m := floor((L + R) / 2)
5.        if S[m] < P:
6.            L := m + 1
7.        else if S[m] > P:
8.            R := m - 1
9.        else:
10.           return true
11.   return false
```

The given array must remain in sorted order before performing a binary search. But we can still find an element in an unsorted array using binary search algorithm, depending on the array and the element we are looking for. For example, Let an array **S[ ] = {2,3,1}**, we are looking for **P = 3**, using binary search algorithm. Here, **L = 1**, **R = 3**, **m = floor((1+3)/2) = 2**. Since **S[2] = 3**, binary search will return **true**.

In this problem, you are given two integers **N** and **P**. The array **S** consists of **N** distinct integers from **1** to **N**. You have to find how many permutations of the array **S** will return **true** if we search **P** using the given binary search algorithm.

Input

Input starts with an integer **T (1 ≤ T ≤ 50000)** denoting the number of test cases. Following T test cases will contain two integers **N(1 ≤ N ≤ 10000000)** and **P(1 ≤ P ≤ 10000000)**.

Output

For each test case, print the case number followed by the number of permutations of the array which will return **true** using the given binary search algorithm. Since the answer can be very big, print the answer modulo 1000000007 ($10^9 + 7$). Follow the sample I/O for more clarity.

Sample Input

```
3
1 1
3 2
7 5
```

Output for Sample Input

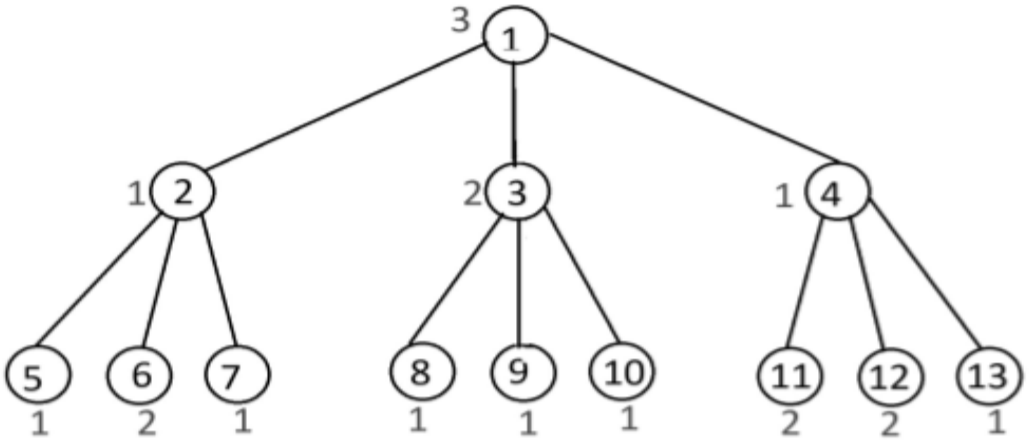
```
Case 1: 1
Case 2: 4
Case 3: 2160
```

Problem D	Input: Standard Input Output: Standard Output
Subtree Query	

You have given a tree of **N** nodes. Nodes are numbered from **1** to **N** and the tree is rooted on the node **1**. Every node of the tree has a value on it.

Subtree rooted at **U** and **V** are similar if they produce the same array after sorting all values of their subtree.

For example,



The number written inside the circle are the node number and number written outside the circle are the value of associative node. (i.e root node has value 3)

Subtree rooted at **2** and **3** are similar, because both of them produce the same array **{1, 1, 1, 2}** after sorting all the values associated with the nodes of those two subtrees. But subtree rooted at **3** and **4** are not similar because subtree rooted at **3** produces the array **{1, 1, 1, 2},** and subtree rooted at **4** produces the array **{1, 1, 2, 2},** and they are not the same array.
Along with the tree you will also given some operation to perform. There are two types of operations

1. **Update(X, V):** add **V** to all the nodes of subtree rooted at **X**.
2. **Query(X, Y):** Check If subtree rooted at **X** and **Y** are similar.

Input

First line of the input will contain a single integer, **T**, denoting the number of test cases. Each test case starts with two space separated integers **N, O**. **N** denotes the number of nodes in the tree and **O** denotes the number of operations.

Next line contains **N** space separated integers, **i-th** of which indicates the associated value of **i-th** node. Next **N-1** lines contain two space separated integer **U, V,** denoting that there is an edge between node **U** and **V**.

Each of next **O** lines contains three integers, describing an operation. First of them is the type of operation, which can be either **1** or **2.** If it is **1**, then the following two integers are **X, V**, denoting an update operation. Otherwise, the following two integers are **X, Y**, denoting a query operation.

Constraints:

- **1 ≤ T ≤ 20**
- **1 ≤ N ≤ 2 x 10^5**
- **-10^5 ≤** value of each node **10^5**
- **1 ≤ O ≤ 2 x 10^5**
- **1 ≤ U, V ≤ N**
- **1 ≤ X, Y ≤ N**
- **-10^5 ≤ V ≤ 10^5**

Summation of **N** over all test cases will be less or equal to **5 x 10^5**. Summation of **O** over all test cases will be less or equal to **5 x 10^5**.

Output

For each test case, print the line without quotation **"Case t:"** Where **t** is the number of the test case starting from **1**. For each **Query(X, Y)** operation print **1** if subtree rooted at **X, Y** are similar; otherwise print **0**, in separate line. Check sample input output for exact formatting.

Sample Input

```
2
13 4
3 1 2 1 1 2 1 1 1 1 2 2 1
1 2
1 3
1 4
2 5
2 6
2 7
3 8
3 9
3 10
4 11
4 12
4 13
2 2 3
2 3 4
1 12 -1
2 3 4
2 1
1 1
1 2
2 1 2
```

Output for Sample Input

```
Case 1:
1
0
1
Case 2:
0
```

N.B.: Input file is huge. Please use faster I/O.

Problem E	Input: Standard Input Output: Standard Output
Balloon Heads	

Have you ever heard of balloon heads? Well, they had a fever and their heads have turned to balloons. As their caretaker, Alice wants to shoot them to put some senses into their (balloon) heads.

There are **n** balloon heads floating on 2d grid. They can be considered as points on the plane. You are given the coordinates of them at time **0**. At the beginning of each second, a balloon head goes up one unit. That is, it goes from **(x, y)** to **(x, y+1)**. But if it's y-coordinate becomes a multiple of **h** then it stays still for **extra t** seconds.

For example, if **h = 5** and **t = 3**, then a balloon head starting at **(3, 8)** at second **0**, will go to **(3, 9)** at second **1, (3, 10)** at second **2**, and then it stays at **(3, 10)** for another **3** seconds (at second **3, 4** and **5**). Then at **6th** second, it will move to **(3, 11)**.

To put the balloon heads to sleep, Alice has decided to perform **q** operations. i'th of her operation can be one of the following two types:

1 T_i : Here T_i is used to generate 4 more parameters X_i , Y_i , DX_i , DY_i . Pseudocode to generate them is mentioned below:

seed := T_i
seed := (seed * 1103515245 + 1234) % 100000001 , X_i := (seed%lim)+1
seed := (seed * 1103515245 + 1234) % 100000001 , Y_i := (seed%lim)+1
seed := (seed * 1103515245 + 1234) % 100000001 , DX_i := (100000000 - (seed%lim))
seed := (seed * 1103515245 + 1234) % 100000001 , DY_i := (100000000 - (seed%lim))

Value of variable **lim** in the above pseudocode is given in input before all queries, which is the same for all queries.

This means at **T_i-th** second, Alice shoots all the points inside and on the border of the rectangle having opposite corners at points **(X_i , Y_i)** and **(X_i + DX_i , Y_i + DY_i)**. It is guaranteed that for the ith and jth query, if **i < j** then **T_i < T_j** . If a balloon head moves at T_i-th second, then its position after the move will be considered for this operation.

2 B_i : This is a query Alice makes to you. The query asks how many times the **B_i-**th balloon head has been shot before this query is made. The balloons are indexed from 1 to n in the order they are listed in input.

Input

The first line of the input contains an integer **T** which denotes number of test cases. Description of T test cases follows.
The first line of each test case has two integers **n** and **q**, which indicates the total number of balloon heads and number of operations Alice performs respectively. The second line of the test case has two integers **h** and **t**, which indicates the constants as mentioned in the statement.
Next line has four integers **sx, sy, dsx, dsy .** From these four numbers, 2n values are generated using the following pseudocode:

```
seed := sx
i := 1
WHILE i ≤ n:
        x_i := sx + (seed % (dsx-sx+1)) , seed := (seed * 1103515245 + 1234) % 100000001
        y_i := sy + (seed % (dsy-sy+1)) , seed := (seed * 1103515245 + 1234) % 100000001
                IF y_i % h > 0 THEN
                i := i+1
        ENDIF
ENDWHILE
```

Here $\mathbf{x_i}$, $\mathbf{y_i}$ indicate coordinate at time **0** of **i-th** balloon head. It is guaranteed that $\mathbf{y_i}$ is not a multiple of **h**.
Next line contains an integer **lim** which will be used to generate query parameters as specified in the statement.

Each of the next q lines indicates one of Alice's operations as mentioned in the statement. It is guaranteed that there will be at least one query of the second type.

Constraints

- **1 ≤ T ≤ 13**
- **1 ≤ n ≤ 200,000, summation of n over all test cases does not exceed 800,000**
- **1 ≤ q ≤ 200,000, summation of q over all test cases does not exceed 800,000**
- **2 ≤ h ≤ 5**
- **1 ≤ t ≤ 5**
- **1 ≤ sx, sy, dsx, dsy ≤ 100,000,000 , sx ≤ dsx, sy ≤ dsy**
- **1 ≤ x_i , y_i ≤ 100,000,000 , y_i is not divisible by h**
- **1 ≤ lim ≤ 100,000,000**
- **1 ≤ T_i , X_i , Y_i ,DX_i ,DY_i ≤ 100,000,000**
- **1 ≤ B_i ≤n**

Output

The first line of each test case should contain the case number in the format: **"Case TC:"** where **TC** is the case number (without quotes).Then, for each of the 2nd type queries, output it's answer in one line. Follow the sample output for more clarity.

Sample Input	Output for Sample Input
1 3 6 2 1 40000000 40000000 60000000 60000000 100000000 1 1 2 1 1 3 2 2 1 8 2 3	Case 1: 0 1 1

N.B.: Input file is huge. Please use faster I/O.

Problem F	Input: Standard Input Output: Standard Output
Mathematical Marathon	

The city of Alexa hosts a marathon every year. The rules of the marathon are simple, the course consists of a series of **N** connected checkpoints joined by **M** roads. Each person starts from a starting checkpoint and must reach a designated destination checkpoint. Now people need water to run. For each litre of water consumed, a runner can run 1 kilometer. Each person starts by consuming **X** litres of water at the starting-point. Water fountains are available at every checkpoint but there are no water fountains in the middle of the roads.

However this year the problem solver guild's annual olympiad was scheduled on the same day as the marathon. During this marathon most of the city remains closed, but the people from the guild are refusing to cancel their event. After much chaos from the guild the city council has agreed to combine the olympiad with the marathon. Each person running the marathon will now have to solve problems while running.

Each runner will be given a maximum capacity of water that he can drink. This amount will be given before the race starts and it will be same for every runner. Initially this capacity is **X**, but after a runner uses a road of length **K**, his new capacity becomes $\mathbf{X_{new} = gcd(X_{old}, K)}$ so he can only consume $\mathbf{X_{new}}$ amount of water at that checkpoint. A runner will not drink more than exactly what is needed to cross a road, even if the maximum capacity allows it, as the extra weight may slow him down. and as there are no checkpoints in the middle of the road, so if a road has length $\mathbf{K > X_{new}}$ he cannot use that road. Given these constraints, the runners have come to you for help. Help them find the shortest route for their race.

Input

The first line of the input is an integer **T**, denoting the number of test cases. Each test case starts with a line consisting of 2 integers **N** and **M** which indicate the number of checkpoints and connecting roads respectively. **i-th** of the following **M** lines contain 3 integers $\mathbf{U_i, V_i, W_i}$. Here $\mathbf{U_i, V_i}$ indicates the 2 ends of an undirected road, and $\mathbf{W_i}$ indicates it's length. There may be multiple roads connecting the same checkpoints. On the last line for the test case, you are given 3 integers, the starting checkpoint **ST**, destination checkpoint **EN**, and the initial capacity **X**.

Output

For each test case, print the case number followed by a single integer denoting the length of the shortest path from the source to the destination, or 'impossible' if there are no valid paths. See sample output for clarification.

Constraints:

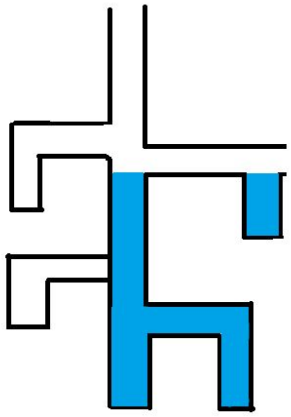
- $\mathbf{1 \le T \le 10}$
- $\mathbf{1 \le N, M, X, W_i \le 10^5}$
- $\mathbf{1 \le ST, EN, U_i, V_i \le N}$

Sample Input	Output for Sample Input
2 6 7 1 2 4 1 4 1 2 3 2 2 4 4 3 6 4 4 5 4 5 6 4 1 6 12 5 5 1 2 4 2 3 1 3 5 3 1 4 6 4 5 10 1 5 12	Case 1: 16 Case 2: impossible

N.B.: Input is huge, please use faster I/O methods.

Problem G	Input: Standard Input Output: Standard Output
Water Puzzle	

We all are familiar with a puzzle, where some pipe is connected with other. Sometimes connections are blocked and sometimes pipes have leakage. When we pour water from top, some part of pipes may have water and some part of pipes may not have any water because of leakage. Below figure is an example.



In the figure above, if we pour water from the top, after a certain amount of time water will leak from the right side. In this problem you will be given a similar structure as a grid with N rows and M columns. The input grid will consist of only '.' and '#' where '.' means empty cell and '#' means blocked cell. Each empty cell can store 1 unit of water.

To clarify more:

- At each unit time one unit water will fall from the top of M columns and due to gravity water will always try to move downwards until it finds any barrier.
- If it finds a barrier, water will be distributed evenly in both directions and will move to surface having lower height due to gravity.
- So, the unit water may fall from the structure due to leakage or it may be stored in some surface bounded by blocks and the height of water level will increase.
- You can simply assume, any nonempty cell as barrier. Also, leaks are so powerful that they can release any amount of water at a single time.
- So, it is guaranteed that after a certain period of time, the amount of water in the structure will become stable(as it is shown in the above figure), that is, no more water will be stored in the structure (water will fall due to leakage or will overflow).

You have to find out the amount of water that will be stored in the structure after being stable.

Input

Input begins with an integer **T**, denotes the number of test cases. Each case begins with two integers **N** and **M** denoting the number of rows and number of columns. Then, there will be N lines with M characters in each line. There will be only '.' and '#' in the grid.

Constraints:
1 ≤ T ≤ 10, 5 ≤ N ≤ 1000, 5 ≤ M ≤ 1000
ΣN*M over all test cases ≤ 2,000,000

Output

For each case of input print one line with format **"Case t: X"**. **t** is the case number starting from 1 and X is the maximum amount of water the structure may contain.

Sample Input

```
2
14 13
....#.#......
#####.#......
#.....#######
#.###........
#.#.#.####.##
###.#.#..#.#.
....#.#..###.
....#.#......
....#.######.
....#......#.
....#.####.#.
....#.#..#.#.
....#.#..#.#.
....###..###.
15 13
....#.#......
#.###.#......
#.....#######
#.###........
#.#.#.####.##
###.#.#..#.#.
....#.#..###.
....#.#......
#.###.######.
#...#......#.
#.###.###..#.
#.#.#.#..#.#.
###.#.#..###.
....#.#..#...
....########.
```

Output for Sample Input

```
Case 1: 19
Case 2: 22
```

Problem H	Input: Standard Input Output: Standard Output
State Cup Madness	

In Australia there are **6** states, Western Australia, Queensland, South Australia, New South Wales, Victoria and Tasmania. Regular sporting events are being arranged among these states.

Currently a cricket league is going on. Mr. X is a big fan of Tasmania. He is very eager to know what is the best possible outcome for his team optimistically. He hired you for doing this. As you are a great programmer, you solved this problem instantly.

But another football league is still going on. Some of the states had taken part in this tournament with different franchise names. Each state could have no more than one team. Some states might not even participate in this tournament. But Tasmania has a team for sure in this tournament. So, Mr. X has appointed you to find out the best possible result for the Tasmanian team.

Match Rules:

- A match is played between two teams.
- Each team score some goals. The team that scores maximum goals wins the match. If they both score the same amount of goals, then it's a draw.
- A win gives you **3** points, draw gives you **1** point and losing gives you **0** point.

Tournament Rules:

- Each team will play a match with other teams in their home.
- So, a pair of teams will face each other in home-away basis.
- Each match is independent. Result of home match will not affect the away match.
- If there are **N** teams in the tournament, a team will play **2** $\times$ **(N - 1)** matches. So, there will be total **N** $\times$ **(N - 1)** matches.
- Teams are sorted according to their total amount of points, the more the better.
- If two or more teams have the same amount of points, they are sorted by their amount of (goal scored - goal conceded), the more the better.
- Still, If two or more teams have the same place, they are sorted by their amount goal scored, the more the better.
- Even if two or more teams have the same place, they are sorted by their franchise names, lexicographically smaller is better.

Already some matches have been played. The scorecard of those match are given to you. You have to find the maximum place that the Tasmanian team can possibly achieve and suitable point difference with the top most other team. If, the tasmanian team can be champion, then they want to maximize the point difference with the second team. Otherwise they want to minimize the point difference with the top team.

Input

Input starts with an integer, **T**, denoting the number of test cases. For each test case there will be two integer **N** and **M**, denoting number of teams in that tournament and number of already played matches. The next line contains **N** space separated words denoting the team names, the first team is the Tasmanian team. The next **M** lines contain scorecard for each already played match in this given format:

team1 g1 - g2 team2

Here, **team1** denotes the home team and **team2** denotes the away team. **g1** denotes the number of goals scored by **team1** and **g2** denotes the number of goals scored by **team2**.

Output

For each test case, print a single line in this format:

Case X: P D

Here, **X** denotes the number of the test case. **P** denotes the maximum achievable place for the Tasmanian team and **D** denotes the suitable point difference.

Constraints

1 ≤ T ≤ 1000

2 ≤ N ≤ 6

0 ≤ M < N * (N - 1)

1 ≤ length_of_team_name ≤ 20 and all characters will be lowercase english letter

0 ≤ g1, g2 ≤ 20

It is guaranteed that the Tasmanian team has at least one match left, all team names are distinct. There is no limit for goals scored for upcoming matches (have to be non-negative for sure).

Sample Input	Output for Sample Input
3 2 1 arzentina brajil arzentina 2 - 3 brajil 3 4 arzentina brajil zermany brajil 0 - 7 zermany zermany 4 - 0 arzentina arzentina 2 - 3 brajil arzentina 0 - 2 zermany 3 4 arzentina brajil zermany brajil 0 - 7 zermany zermany 4 - 0 arzentina arzentina 2 - 2 brajil arzentina 0 - 2 zermany	Case 1: 1 0 Case 2: 2 9 Case 3: 2 5

Problem I	Input: Standard Input Output: Standard Output
King of Africa	

You are the king of the gridland in africa. There are **N** tribes lives in your country. To make the service management of the country you divide the whole country land to **NxN** square grid. To send proper service to every tribe you want to rearrange the country's tribe in a way that no two cell in a row or column will contain same tribe members. In order to do so,

- You can move people from one cell to another cell if they have same tribe. For example, let cell**(x1, y1)** and cell **(x2,y2)** have people of the same tribe. So, if you move people from cell **(x1,y1)** to cell**(x2,y2)** it will left cell **(x1,y1)** as empty.
- You can not move any tribe to any empty cell.

To do this he can just pick **N** cell, one of each tribe and move all the **people** of same tribes to those cells. But this will make the country looks like empty. So he wants rearrange in a way that he need to empty the minimum number of cells. If there are several such ways, he will choose the cells in a way that he needs to move the minimum number of people.

As you are the best friend and cabinet members of the king so he asked you to help him to achieve the constraints.

Input

First line, there will be an integer **T (1≤T≤100)** which is the number of test cases.

Every test case starts with a number 1**≤N ≤ 100**.

Next **N** line contain **N** integer which denoes **NxN** grid **A**. where element in **i-th** row and **j-th** column **A[i,j]** denotes the tribe number living in that cell.

Next **N** line contain **N** integer which denoes **NxN** grid **B**. where element in **i-th** row and **j-th** column **B[i,j]** denotes the number of people of tribe **A[i,j]** living in that cell.

Constraints

- **1 ≤ A[i, j] ≤ N**
- **1 ≤ B[i, j] ≤ 10^5**

Output

For each test case you need to print a line **"Case t: X Y"**. where **t** is the case number, **X** is the minimum number of cells king need to empty and **Y** is the minimum number of people king need to move.

Sample Input

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

Output for Sample Input

```
Case 1: 2 3
```

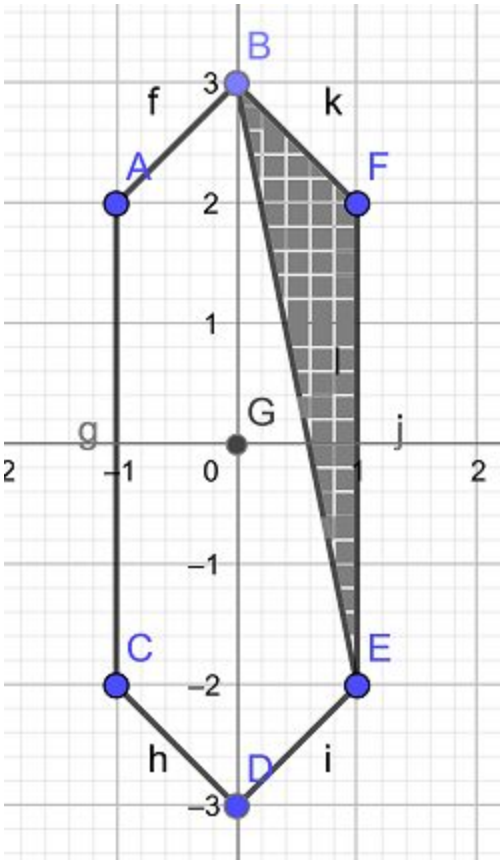
N.B.: Input is huge, please use faster I/O methods.

Problem J

Input: Standard Input
Output: Standard Output

No Points in Polygon

You will be given a convex polygon and several points which lie completely inside this given polygon. You need to make a cut connecting any two vertices of the given polygon such that area of the new polygon that contains none of the given points (*inside or on its edge*) is maximized.



Note: A point is considered to be completely inside a polygon if the point lies strictly within the enclosed area, and not on any of the edges.

Input

First line of input contains an integer **T (1 ≤ T ≤ 20)**, the number of test cases.

For each test case, first line contains two integers, **N** and **M**. Here, **N (3 ≤ N ≤ 3x10^5)** is the number of vertices of the convex polygon, and **M (1 ≤ M ≤ 3x10^5)** is the number of points given inside the polygon.

Each of the following **N** lines contains two integers **(x_i, y_i)**, describing a vertex of the given convex polygon. Vertices will be given in a counter clockwise order.

Finally, each of the following **M** lines contains two integers **(x_j, y_j)**, describing a point that lies within the given convex polygon.

For all coordinates **(x, y)**, it is given that **$|x|, |y| \leq 10^9$**. And, it is guaranteed that the given convex polygon will always have a positive area.

Output

For each test case, print a line containing the test case number starting from **1** and an integer **X**, where **X = TWICE** the maximum possible area as described in the problem statement. Check sample input and output for details.

Sample Input

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

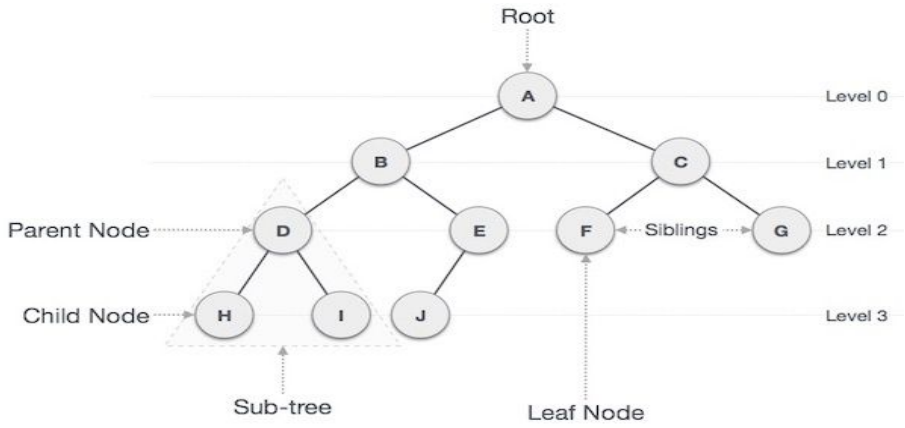
Output for Sample Input

```
Case 1: 4
Case 2: 4
```

N.B.: Input file is huge. Please use faster I/O.

Problem K	Input: Standard Input Output: Standard Output
Depth of Binary Tree	

Tree is a data structure where the nodes are connected by edges. In each tree, there will be a node called the root and every node is connected to the root directly or indirectly. All the nodes will have zero or more nodes connected to it directly as it's children. The depth of a tree is denoted by the highest level in a tree. Which is the maximum distance for any node from root of that tree. A binary tree is a tree in which each node has at most two children, which are referred to as the left child and the right child. The following picture contains a binary tree with depth = 3.



You will be given **N,** which denotes the total number of nodes in a binary tree. You have to print what is the maximum and minimum possible depth a binary tree can have if it has **N** nodes.

Input

The first line of the input contains an integer **T(1 ≤ T ≤ 20)**, denoting the number of test cases. Then in next **T** lines, **T** test cases will follow. Each line will have **N(1 ≤ N ≤ 20)**.

Output

For each test case, print the results in *"**Case I: MAX MIN**" format*, where ***I*** is the case number starting from **1** and **MAX, MIN** are maximum and minimum possible depth.

Sample Input

```
3
1
2
3
```

Output for Sample Input

```
Case 1: 0 0
Case 2: 1 1
Case 3: 2 1
```

BUP Inter University Programming Contest 2019
BUP-Techsurgence'19

Hosted by BUP
Dhaka, Bangladesh

6th April 2019

You get 17 Pages, 10 Problems & 300 Minutes

Platform Support:

CodeMarshal

Problem Set By:

Bangladesh Association of Problem Setters

Problem A

Input: Standard Input
Output: Standard Output

Expected Coin Change (II)

Mr. Kiptus has a bag of **N** coins. These are **N** different kinds of coins with value from **1** to **N** Taka; one coin of each type (he is rich). He wants to donate some coins. He has decided not to donate more than **M** Taka (he is also kiptus). But he is also interested to keep things dynamic. So he devises the following procedure:

1. He will first start with 0 donations.
2. He will take a coin randomly from the bag. The probability of getting any coin left in the bag is same. If there is no coin left, he will stop donating.
3. He will then check if on donating this coin, his total donation exceeds M or not. If not, then he will donate the coin, and his donation will increase by the coin's value and start again from step 2. Otherwise, he will return the coin to the bag and stop donating.

Now he is wondering what is the expected amount he will donate. Can you calculate it?

Input

The first line of the input is **T (T ≤ 40)**, then T test cases follow. In each case, there will be two integers **N (1 ≤ N ≤ 30)** and **M (0 ≤ M ≤ 10^7)**.

Output

For each test case, print a line of the format "**Case I: A/B**", where **I** is the case number and **A/B** is the expected money Mr. Kiptus will donate in reduced fraction form (A and B are coprime). It is guaranteed that **A** and **B** will be less than **10^{16}**.

Sample Input	Output for Sample Input
3 3 5 5 3 10 10	Case 1: 4/1 Case 2: 27/20 Case 3: 3127/420

Tips: For 75% test cases, N ≤ 25.

Problem B

Input: Standard Input
Output: Standard Output

Deja Vu

In this problem, all you have to do is generate an array having **N** positive integers which satisfies **Q** constraints.

Every constraint is of the following kind - "The least common multiple of all the numbers from the **L**-th position to the **R**-th position of the array has to be a multiple of **X**".

You need to find the lexicographically smallest array which satisfies all the constraints.

Input

The first line of the input contains an integer **T**, denoting the number of test cases. Then the description of the **T** test cases follow. The first line of every test case will contain two integers **N** and **Q**, denoting the length of the array and the number of constraints respectively. Each of the following **Q** lines will contain a constraint of the following form: **L R X**.

Constraints

- **$1 \le T \le 100$**
- **$1 \le N \le 10^5$**
- **$1 \le Q \le 10^5$**
- **$1 \le L \le R \le N$**
- **$1 \le X \le 10^6$**

Both, sum of N and sum of Q over all test cases $\le 4 \times 10^5$

Output

For each test case, print a line containing the case number and the lexicographically smallest array which satisfies all the given constraints. As the numbers of the array can be very large, output the numbers modulo **1000000007 ($10^9 + 7$)**.

Sample Input

```
1
7 2
2 5 2
5 7 3
```

Output for Sample Input

```
Case 1: 1 1 1 1 2 1 3
```

An array **C** of size **N** is called lexicographically smaller than another array **D** of equal size, if and only if there exists an **$i \le N$** such that:

C[p] == D[p] for all **$1 \le p < i$** and **C[i] < D[i]**

Warning: Dataset is huge. Please use faster I/O methods.

Problem C

Input: Standard Input
Output: Standard Output

Theory of Compression

Big data is a buzzword, isn't it? According to Wikipedia *"Big data refers to data sets that are too large or complex for traditional data-processing".* A gigantic amount of data is generated from different sources like news portals, social media, blogs, communication, photos, services and so on. On average 2.5 quintillion bytes of data are created each day. Large data servers are used to store these data and it is ever growing. So, we need a more efficient compression technique for storing data.

Mr. Maddy, a mad scientist came up with a new algorithm for text data compression named "theory of compression". The theory is as follows:

- **T** is the original text and **C** is the compressed text, where $\mathbf{C = P_1R_1P_2R_2P_3R_3\ldots P_nR_n}$
- The tuple { $\mathbf{P_i}$, $\mathbf{R_i}$ } means the pattern $\mathbf{P_i}$ is repeated $\mathbf{R_i}$ times in the original text.That is $\mathbf{P_1}$ is repeated $\mathbf{R_1}$ times, $\mathbf{P_2}$ is repeated $\mathbf{R_2}$ times … $\mathbf{P_n}$ is repeated $\mathbf{R_n}$ times consecutively.
- $\mathbf{P_1, P_2, P_3, \ldots , P_n}$ are composed of lower case Latin letters **(a-z)** only and $\mathbf{1 \le |P_1|, |P_2|, \ldots , |P_n| \le 10^5}$
- $\mathbf{1 \le R_1, R_2, R_3, \ldots , R_n \le 50000}$ and they do not contain any leading zeros.

For example, let **T = ababcccddde**. Here, **C = ab2c3d3e1** is the compressed text where,
$\mathbf{P_1}$ **= ab**, $\mathbf{R_1}$ **= 2**, $\mathbf{P_2}$ **= c**, $\mathbf{R_2}$ **= 3**, $\mathbf{P_3}$ **= d**, $\mathbf{R_3}$ **= 3**, $\mathbf{P_4}$ **= e**, $\mathbf{R_4}$ **= 1**. If we decompress **C** we will get the text **T**.

In this problem, you will be given **C** and two integers **l** and **r** ($\mathbf{1 \le l, r \le 10^{12}}$ and $\mathbf{0 \le r - l < 10^4}$). You have to decompress **C** into **T** and print $\mathbf{T_lT_{l+1}T_{l+2}\ldots T_r}$. That is, you have to print the sequence **T[l...r]** from the indices **l** to **r** inclusive (1 based index).
For the above example **C = ab2c3d3e1**. So, **T = ababcccddde.**

- If, l = 2, r = 5, the answer will be **babc**.
- If, l = 9, r = 9, the answer will be **d.**
- If, l = 4, r = 15, the answer is **-1** as there is no such valid sequence.

Input

Input starts with an integer **TC** ($\mathbf{1 \le TC \le 10}$) denoting the number of test cases. Each of the test cases starts with a non-empty string **C** ($\mathbf{1 \le |C| \le 2 \times 10^5}$) denoting the compressed text. Next line of each test case will contain an integer **Q** ($\mathbf{1 \le Q \le 200}$) denoting the number of queries. Next, there will be **Q** lines each containing two integers **l** and **r** ($\mathbf{1 \le l, r \le 10^{12}}$ and $\mathbf{0 \le r - l < 10^4}$) . It is guaranteed that inputs are valid and $\sum$ (r - l +1) in all test cases will be less than $\mathbf{10^7}$.

Output

For each test case, first, print the test case number in a single line and print the sequence **T[l...r]** of every query in a separate line. If there is no such valid sequence, print **-1** instead. Check the sample I/O section for more clarity.

Sample Input

```
2
ab2c3d3e1
3
2 5
9 9
4 15
a10
2
1 10
1 11
```

Output for Sample Input

```
Case 1:
babc
d
-1
Case 2:
aaaaaaaaaa
-1
```

Warning: Dataset is huge. Please use faster I/O methods.

Problem D

Input: Standard Input
Output: Standard Output

Yatzy

Yatzy is a very popular game. Since the actual rules of the game and rules for this problem are not exactly the same, you are requested to go through all the rules very carefully.

Here are the rules of the game:

- You have **5** regular dice. Each of the dice has **6** faces and these faces contain **1** to **6** exactly once and each outcome is equiprobable. Outcomes of the dice are independent from each other.
- The game is played in **12** rounds.
- In each round, there are **3** stages.
- In the first stage, you will roll all the **5** dice.
- In the second stage, you can lock some of your dice(possibly none and all). If you lock some dice, you can not roll this in the current stage and the outcome of that dice remains same as the previous stage. After locking some dice, you will roll the dice that are not locked.
- In the third stage, you will do the same as the second stage.
- Now you have the final combination of the dice. The order of the dice does not matter. Here is an example of **3** stages: In the the first stage, you get **12135**. You lock the first and third (from left) dice and roll again. Then you get **14146**. Now you lock the second and fourth dice and roll again. Then you get **34246**. So, this is your final combination. You can treat this combination as **23446**, **64432**, **44632** etc too.
- There are **12** scoring zones. When you get your final combination, you have to choose a scoring zone. You will get some points according to your dice combination and chosen scoring zone.
- In each round you can choose only one scoring zone. And you can not choose a scoring zone more than once. So, basically after **12** rounds you choose each of those scoring zones exactly once.

Here is the point system for each scoring zone:

- **Scoring zone 1:** You get the sum of the outcome of those dice where the outcome is **1**. Example: If the combination is **11234**, you get **2** points. You will consider only those dice who have the outcome **1**. Then you get the sum of the outcome of those dice. So, you get **1 + 1 = 2** points.
- **Scoring zone 2:** You get the sum of the outcome of those dice where the outcome is **2**. Example: If the combination is **12224**, you get **6** points. You will consider only those dice who have the outcome **2**. Then you get the sum of the outcome of those dice. So, you get **2 + 2 + 2 = 6** points.
- **Scoring zone 3:** You get the sum of the outcome of those dice where the outcome is **3**.
- **Scoring zone 4:** You get the sum of the outcome of those dice where the outcome is **4**.
- **Scoring zone 5:** You get the sum of the outcome of those dice where the outcome is **5**.
- **Scoring zone 6:** You get the sum of the outcome of those dice where the outcome is **6**.
- **Scoring zone 7:** It is called **3 of a kind**. If you have at least **3** dice with the same outcome, then you get the sum of all the **5** outcomes. Otherwise you get **0** point. Example: If the combination is **33346**, you get **19** points.
- **Scoring zone 8:** It is called **4 of a kind**. If you have at least **4** dice with the same outcome, then you get the sum of all the **5** outcomes. Otherwise you get **0** point. Example: If the combination is **33336**, you get **18** points.

- **Scoring zone 9:** It is called **full house**. If you have exactly **3** dice with the same outcome and **2** dice with some other same outcome, then you get **25** points. Otherwise you get **0** point. Example: If the combination is **22255**, you get **25** points. But if the combination is **22222**, you get **0** point.
- **Scoring zone 10:** It is called **small straight**. If you get at least **4** outcome that forms an arithmetic series with difference **1**, then you will get **30** points. Otherwise you get **0** point. Example: If the combination is **34561**, you get **30** points.
- **Scoring zone 11:** It is called **large straight**. If you get all **5** outcome that forms an arithmetic series with difference **1**, then you will get **40** points. Otherwise you get **0** point. Example: If the combination is **12345** or **23456**, you get **40** points.
- **Scoring zone 12:** It is called **yatzy**. If you have exactly **5** dice with the same outcome, you get **50** points. Otherwise you get **0** point. Example: If the combination is **11111** or **33333**, you get **50** points.

You will be given some intermediate state of the game. You have to find the maximum possible expected points you can get at the end of all the **12** rounds.

Input

Input starts with an integer **T (1 $\leq$ T $\leq$ 1000)**, denoting number of test cases.

For each test case, in the first line there will be **12** space separated integers denoting the already earned points in each scoring zone. The integers will be given serially according to the problem statement. If the point is **-1** in some scoring zone, that means it has not been chosen yet. You can be sure that the already chosen scoring zones will have valid points assigned to them and there will be at least one scoring zone that has not been chosen yet.

In the next line, there will be an integer **S (1 $\leq$ S $\leq$ 3)**, denoting the stage you are about to play. If **S = 1**, you are about to play the first stage and no dice is rolled yet. If **S = 2** or **S = 3**, you are in the second or the third stage and you already have a combination and have to roll again.

In the next line, there will be **5** space separated integers denoting the outcome of the dice in the previous stage. If **S = 1**, all the outcome of the dice will be **-1** as there is no previous stage.

Output

For each test case, print a line in this format, **Case X: Y**. Where **X** denotes the test case number and **Y** denotes the maximum possible expected points.

Your answer will be accepted if its relative or absolute error does not exceed 10^{-6}.

Mathematically, if your answer is **A** and the jury's answer is **B**, then your answer is accepted if and only if $\frac{|A-B|}{max(1,|B|)} \leq 10^{-6}$.

Sample Input

```
3
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
1
-1 -1 -1 -1 -1
0 0 0 0 0 0 0 0 0 0 0 -1
1
-1 -1 -1 -1 -1
0 0 0 0 0 0 0 0 0 0 0 -1
3
1 1 1 1 2
```

Output for Sample Input

```
Case 1: 200.469963431
Case 2: 2.301432126
Case 3: 8.333333333
```

Problem E

Input: Standard Input
Output: Standard Output

Paint The Doors

There are **N** students in a classroom. Each students has a unique id **1, 2, 3, … , N**. There are **M** doors in the school which are labeled as **1, 2, … , M**. The doors need to be painted. Each student is assigned to paint some doors. The **i-th** student is assigned with a range L_i to R_i and he/she will paint all the doors whose labels are between L_i to R_i inclusive. It may happen that a single door is painted by multiple students.

Now you will be given **Q** queries to answer. Each query will contain four integers L_M R_M L_N R_N. Now you have to answer how many doors between label L_M and R_M are there which are painted by at least one student with id between L_N and R_N inclusive.

Input

First line of the input contains a single integer **T (T ≤ 5)** denoting the number of test cases. Each test case starts with a line containing two integers **N (1 ≤ N ≤ 10^5)** and M **(1 ≤ M ≤ 10^5)**. Each of the next **N** lines will contain two integers L_i and R_i **(1 ≤ L_i ≤ R_i ≤ M)** denoting the range associated with the i-th student. Next line will contain a single integer **Q (1 ≤ Q ≤ 10^5)** denoting the number of queries. Each of the next **Q** lines will contain four integers L_M R_M L_N R_N (**1 ≤ L_M ≤ R_M ≤ M** and **1 ≤ L_N ≤ R_N ≤ N**).

Output

The first line of each test case should contain the case number in the format: "**Case X:**" (without quotes) where **X** denotes the test case number. After that **Q** lines follow each containing the answer of the queries.

Sample Input	Output for Sample Input
1 2 10 1 5 6 10 4 3 5 1 2 3 5 2 2 4 7 1 2 4 7 1 1	Case 1: 3 0 4 2

Problem F

Input: Standard Input
Output: Standard Output

Winter is Coming

Winter is coming and the seven kingdoms must prepare for what it brings. The army of white walkers is marching closer, led by the Night King. Apocalypse is near and It looks like the end of the world. The Starks, Lannisters and Greyjoys must put aside their hatred for each other and bond together if the white walkers are to be defeated.

There are **N** strongholds in the seven kingdom, each of which must be occupied by the force of one of the three houses: House Stark, House Lannister, and House Greyjoy. No two houses can occupy the same stronghold. Also, there are **M** tunnels connecting a pair of different strongholds. A tunnel connecting strongholds can be used in both ways. For strategic reasons, the leadership agrees that it is not wise to place forces from the same house on two strongholds connected by a tunnel.

But alas, it is not easy to find such an assignment. There is little time to spare, and winter is coming. The kingdom needs to find a valid assignment of the houses to the strongholds before the white walkers attack. Given a description of the strongholds and the tunnels, you need to find out whether it is possible to assign houses to all the strongholds satisfying all the constraints mentioned above.

Input

The very first line contains an integer **T**, denoting the number of test cases.

Each test case starts with two integers **N** and **M**, denoting the number of strongholds and the number of tunnels connecting strongholds respectively. This is followed by a line containing the names of **N** different strongholds separated by a single space. Names will contain only alphanumeric characters, are case sensitive, will be distinct and won't consist of more than **30** characters. The following **M** lines will contain the description of the tunnels, each of these line will have names of two different strongholds mentioned before, implying they are connected via a tunnel.

There will be a blank line before the start of every test case.

Constraints

- **1 ≤ T ≤ 30**
- **1 ≤ N ≤ 30**
- **1 ≤ M ≤ 330**
- **1 ≤ |Names| ≤ 30**

Output

For each test case, first print the case number (starting from **1**), followed by "**Valar Morghulis**" if it is not possible to assign houses to all the strongholds and the kingdom is doomed. Otherwise, print "**Valar Dohaeris**" after the case number if there is hope. Refer to the sample i/o section for further details.

Sample Input	Output for Sample Input
3 3 3 Winterfell KingsLanding TheWall Winterfell KingsLanding Winterfell TheWall KingsLanding TheWall 6 10 KingsLanding Dorne Braavos CasterlyRock Dragonstone IronIslands Dorne Braavos Braavos CasterlyRock CasterlyRock Dragonstone Dragonstone IronIslands IronIslands Dorne KingsLanding Dorne KingsLanding Braavos KingsLanding CasterlyRock KingsLanding Dragonstone KingsLanding IronIslands 2 1 Mistwood Cornfield Mistwood Cornfield	Case 1: Valar Dohaeris Case 2: Valar Morghulis Case 3: Valar Dohaeris

Problem G

Input: Standard Input
Output: Standard Output

A Thousand Push Ups

Jake and Rosa are best buddies. They work in the police force of a particular precinct of a particular city of a particular country. They have investigated many cases as partners. After solving each case, they have to go to the city of the HQ(headquarter) from their own city to submit the report. For going to the HQ, Rosa always drives her motorbike and Jake rides with her. Rosa loves her bike. But the thing is, Jake hates riding on her bike more than anything! From the HQ, they travel back by train and Rosa's bike is transported as cargo on the train.

One day, Jake was saying that he dreaded riding on Rosa's bike so much that for many roads in the country, he remembered **at least** how many times they have used them for going to the HQ after solving cases. He also said that even for many cities, he remembered **at least** how many times they have visited those cities while going to the HQ after solving cases. In reply, Rosa said that she remembered **at most** how many times they have used many particular roads and also **at most** how many times they have visited many particular cities while going to the HQ after solving cases. Now based on this, Savant, a hacker who works in their precinct said that he could figure out **the minimum** and **the maximum** number of cases Jake and Rosa could have solved together. So he asked for two lists from Jake and Rosa. Since Savant is a weird guy but is also a brilliant hacker, Jake and Rosa did not waste any more time asking how he was going to do that and instead created their lists and gave him.

Formally there are **N** cities in the country and **M unidirectional** roads. The cities in the country are numbered from **1 to N**. Jake and Rosa live in the city numbered **1**. The HQ is placed in the city numbered **N**. In this country, from any city **u**, it is **impossible to return to u** visiting other cities by road.

In Jake's list, there are **A** items in one of the following formats:
0 e x (meaning the *e-th* road is used **at least x** times)
1 u x (meaning the *u-th* city is visited **at least x** times)

In Rosa's list, there are **B** items in one of the following formats:
0 e x (meaning the *e-th* road is used **at most x** times)
1 u x (meaning the *u-th* city is visited **at most x** times)

Now your task is to do what Savant said he could do. Find out **the minimum** and **the maximum** number of cases they could have solved! Oh, there is one more thing that Savant knows, which is, Jake and Rosa have not solved more than 10^9 cases. So even if it is possible in a situation that they can solve more than 10^9 cases, Savant would report the maximum number of cases to be 10^9.

Input

There are **T** test cases. For each case, in the first line, there are two integers, **N** and **M**. **M** lines will follow, each containing two integers **a** and **b** denoting there is a road from the *a-th* city to the *b-th* city. The roads are numbered as given in the input with **1-based indexing**. After that, the next line will contain one integer, **A**. Then there will be **A** lines where each line will contain three integers representing an item of Jake's list. Then the next line will contain the integer, **B**. Then there will be **B** lines where each line will contain three integers representing an item of Rosa's list.

Constraints

- **1 ≤ T ≤ 100**
- For **80%** of the test cases, **2 ≤ N ≤ 10**
- For **20%** of the test cases, **2 ≤ N ≤ 100**
- $1 \leq M \leq \frac{N \times (N-1)}{2}$
- **0 ≤ A, B < N+M**
- **0 ≤ x** in any item of any list **≤ 10^9**
- **1 ≤ e** in any item of any list **≤ M**
- **2 ≤ u** in any item of any list **≤ N**
- All roads are distinct.
- There is no self-loop.
- No two items in the same list would be about the same city or the same road.
- There is no road that originates from the city of the HQ.

Output

For each case, state the case number and then print the answer for that case. If there is no way that all the items in the two lists are correct, you should just print **"-1"** (without the quotes). Otherwise, if there is at least one way to make sure that all the items in the two lists can be correct, you should print **the minimum** and **the maximum** number of cases Jake and Rosa could have solved together according to their lists.

Sample Input

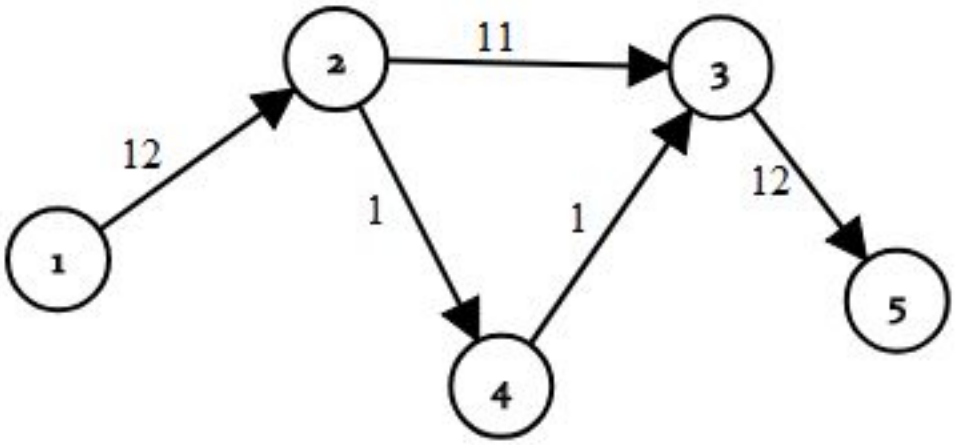
```
1
5 5
1 2
2 3
2 4
4 3
3 5
3
0 1 10
0 4 1
1 2 12
4
0 1 15
0 4 1
0 2 13
1 3 16
```

Output for Sample Input

```
Case 1: 12 14
```

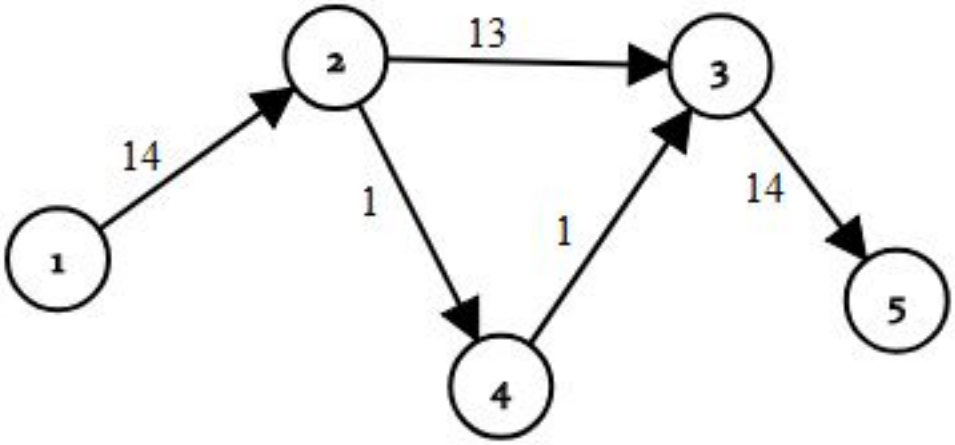
Explanation

A possible way where Jake and Rosa have solved 12 cases is shown below:



Here, each circle represents a city and the number of the city is written inside the circle. The number 11 written above the road from 2 to 3 signifies that after solving 11 cases, they used this road on their route to the HQ. There is no other way to solve less than 12 cases in the given test case.

A possible way where Jake and Rosa have solved 14 cases is shown below:



There is no other way to solve more than 14 cases in the given test case.

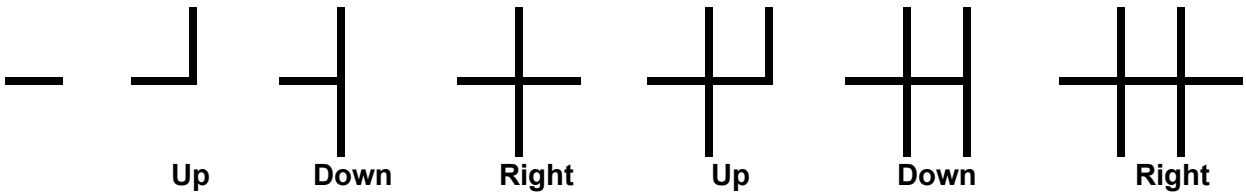
Problem H

Input: Standard Input
Output: Standard Output

Toothpicks

There lives a certain boy somewhere in the world who loves problem solving, spends his time alone thinking about different problems and is obsessed about playing with small objects. Whenever he goes to a restaurant, after ordering food, while people chatter indistinctly, he usually keeps quiet, listening. In the meantime, he subconsciously moves the utensils (plates, forks, spoons, tissue holders etc) on the table so that everything is in a formed manner. That's why some of his close friends refer that he has a little OCD!

One day, in a restaurant, he found a box of toothpicks in his table. He brought some of those and started playing. He placed the toothpicks in a specific order so that each two create a right angle (90 degrees) at their common end. He formed a long connected structure with those, like contiguous plus signs as shown in the image. Suddenly he came up with this problem: 'How many right angles can be formed in this way using **N** toothpicks?' The answer certainly was not **N-1**. The following image (left to right) shows the order in which he would place the first 7 toothpicks. Here, the number of 90 degree angles are 0, 1, 2, 4, 5, 6 and 8 respectively. Notice that, after the first toothpick the order of placement is Up, Down, Right, Up, Down, Right, Up and so on.



While he is busy playing, we want to do the calculations. And by we, we mean you! You will be given the number of toothpicks available. You have to find the number of 90 degree angles formed if those toothpicks are organized according to the boy's idea.

Input

Input will start with a positive integer, **T (≤ 100)** denoting number of test cases. Each case will contain a single integer **N (1 ≤ N ≤ 10000)** in a line denoting the number of toothpicks.

Output

For each case, print the case number first in the format "**Case T:** " without the quotes. Then print an integer denoting the number of 90 degree angles formed. Then print a new line. See sample I/O for clarification.

Sample Input	Output for Sample Input
3 1 2 3	Case 1: 0 Case 2: 1 Case 3: 2

Problem I

Input: Standard Input
Output: Standard Output

Mysterious Lock Maker

Mr. Bokkor is a lock maker. Each key of his locks is a series of alphabet [a-z]. He is called mysterious because he tries to keep the key of the lock from the english dictionary so that people who buys his lock can easily remember the key. Recently, the association of lock maker has introduced some rules for every lock maker in his country. Every lock maker is given a signature. The signature contains only alphabet [a-z]. While making a lock, the maker has to make sure that, the key can be built from the signature after deleting zero or more alphabets from the signature. You can not add or rearrange the alphabets in the signature while making keys.

Now Mr. Bokkor has a serious problem while making key for a lock. Because he chooses the key from english dictionary and needs to check if the key is compatible with the regulation of the association. So he is asking your help to check the validity of keys. He will give you the signature and a dictionary. You need to check if the words of the dictionary is a valid key or not.

Input

First line of the input contains an integer **T ($1 \leq T \leq 100$)**, The number of test case. For every test case first line contains a string S. **($1 \leq |S| \leq 10^5$).** Second line contains an integer **N**, number of words in dictionary. Next N lines contain the words of the dictionary. One word in one line.

Output

For every test case there will be **N+1** line of output.

First line of the output contains a string "**Case X:**", where X is the test case number.

For each of the next **N** lines, print '**valid**' if the i-th word of the dictionary can be used as a key, otherwise print '**invalid**'. See sample I/O for clarification.

Constraints

- **$1 \leq T \leq 100$**
- **$1 \leq |S| \leq 10^5$**
- **Summation of the length of words in all test cases $\leq 2 \times 10^5$**

Sample Input	Output for Sample Input
1 aabcxyza 5 abc abca aaxya aaaa aaxzy	Case 1: valid valid valid invalid invalid

Problem J

Input: Standard Input
Output: Standard Output

Grids and Tiles

There are two grids, **P** and **Q**. Size of grid **P** and **Q** is respectively $u \times v$ and $m \times n$. You need to find out how many $u \times v$ subgrids are there in grid **Q** where $f(X_k, P) \leq S$ (here X_k is the k[th] subgrid of size $u \times v$ of **Q**). Each grid can have two type of tiles, Tile A and Tile B. Each tile has unit length and some shaded area. Let's say the **shaded area** of the $(i, j)^{th}$ tile of grid $\mathbf{X_k}$ and **P** is respectively $\mathbf{X_{k(i, j)}}$ and $\mathbf{P_{(i, j)}}$. Then,

$$f(X_k, P) = (\sum_{i=1}^{u} \sum_{j=1}^{v} |X_{k(i,j)} - P_{(i,j)}|)$$

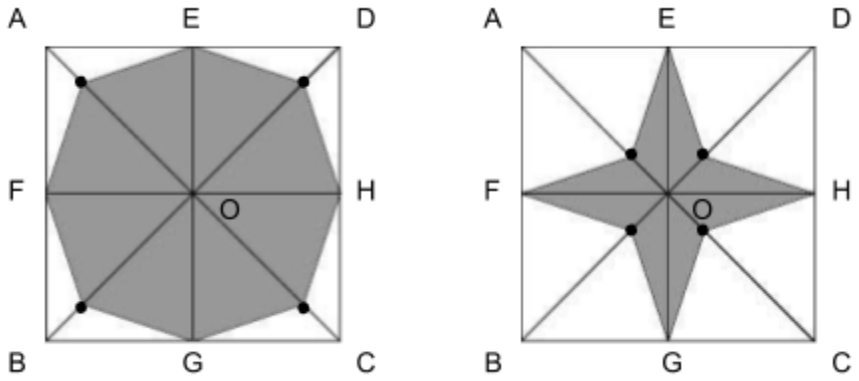


Fig 1: Tile A and Tile B from left to right

All the sides of each shaded area have the same lengths. In both tiles, AC and BD are two diagonals and AE = ED = DH = HC = CG = CB = BF = FA. O is the center of the tiles. In Tile A, the distance between center and each of the **black dotted** corner is $\frac{3AO}{4}$. In Tile B, it's $\frac{AO}{4}$. Two subgrids of size $u \times v$ will be considered different if their starting positions are different.

Input

The first line contains the number of test cases **T (1 ≤ T ≤ 20)**. For each test case, the first line will have five integers **u, v, m, n** and **S(0 ≤ S ≤ 10^5)**. Each of the following **u** lines will have a string of length **v**. Each of the following **m** lines will have a string of length **n**. Each string will have only two type of characters, 'A' & 'B' where 'A' denotes Tile A and 'B' denotes Tile B. **Please use fast I/O**.

Constraints

- **1 ≤ u ≤ m ≤ 1000**
- **1 ≤ v ≤ n ≤ 1000**
- $(\sum_{i=1}^{T} m_i \times n_i) \leq 3 \times 10^6$

Output

For each test case, print the case number and the result of the problem.

Sample Input	Output for Sample Input
2 2 2 3 3 2 AA AA AAA ABA AAA 2 2 3 3 0 AA AA AAA BAA AAA	Case 1: 4 Case 2: 2

DUET Inter University Programming Contest 2019
DUET CSE Fest 2019

Hosted by DUET
Dhaka, Bangladesh

27th April 2019

You get 17 Pages, 9 Problems & 300 Minutes

Platform Support:

CodeMarshal

Problem Set By:

Bangladesh Association of Problem Setters

Problem A

Input: Standard Input
Output: Standard Output

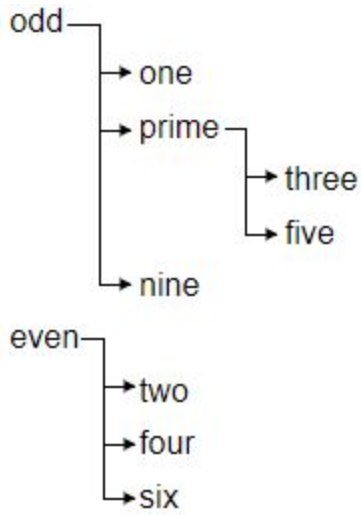
All About Respect

In this problem you will be given a block as input. You need to sort it lexicographically maintaining the block structure and print in sorted order.

A block has the following criteria:

- Block is non-empty.
- A block contains:
  - Elements separated by newlines.
  - Each Element contains:
    - A non-empty string of lower case letter only.
    - Zero or more 'sub-block's. A 'sub-block' will be indented using a **single whitespace character** from the particular element. Each 'sub-block' will have the same properties of a block.

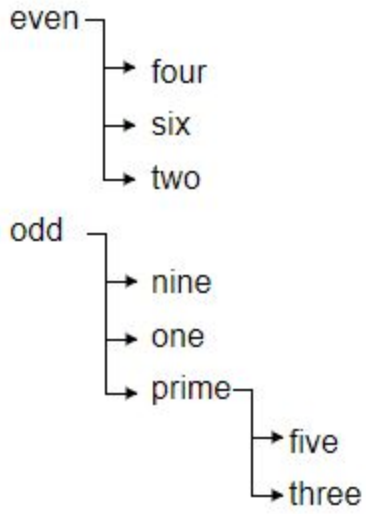
A block of size **N** will have **N** 'new line' separated strings in total. For example, consider the following block of size 10:



(For clear view, the indentations are demonstrated in the above way. IO files will have a single space character for a single indentation, check samples)

The element “odd” contains a single sub-block which contains three elements “one”, “prime” and “nine”. The element “prime” has a sub-block which contains two elements “three” and “five”.
The element “even” contains a single sub-block of three elements “two”, “four” and “six”.

If you simply sort it lexicographically, the logical block structure will be lost. You need to sort the block lexicographically, but respecting the indentations to preserve the block structure. The desired output of the above example:



(For clear view, the indentations are demonstrated in the above way. IO files will have a single space character for a single indentation, check samples)

Input

The first line will contain a single integer **T (1 ≤ T ≤ 100)**. Each test case will have a single integer **N (1 ≤ N ≤ 100)**, denoting the size of the block. The following N lines will contain strings constituting the block. For each test case, all strings are unique.

Output

Print the case number in a single line followed by the block in desired format. Please see sample for details.

Sample Input

```
1
10
odd
 one
 prime
  three
  five
 nine
even
 two
 four
 six
```

Output for Sample Input

```
Case 1:
even
 four
 six
 two
odd
 nine
 one
 prime
  five
  three
```

Problem B

Input: Standard Input
Output: Standard Output

Jumping Gollum

There is an array **A** of **N** integers indexed from **1** to **N**. Initially Gollum is sitting at the **index 1** and the precious ring is at the **index N**. Obviously, Gollum is trying to get the ring as soon as possible.
Let's say, Gollum is at the index **i (1 ≤ i < N)**. Then, in one second Gollum can take any of the following two moves:

- Move 1: Jump to the index **i + 1**
- Move 2: Jump to the index **j** where **i < j ≤ N, gcd(A[i], A[j]) > 1.**

But we know that Gollum is not superman. So there are some limitations on its jumping capability. In one jump Gollum can not move more than **K** index forward, that means if Gollum is at the index **i**, it can reach the index **j** if the following condition holds: **i < j ≤ i + K.**

You have to find the minimum time (in seconds) needed to reach the **index N,** so that Gollum can take the precious ring.

Input

The first line of the input is an integer **T**, denoting the number of test cases. Next, there will be **2T** lines describing the **T** test cases. There are exactly two lines for describing a test case. The first of them contains two space-separated integers **N and K.** The second line contains **N** space separated Integers which represent the array **A**.

Output

For each test case, print a line in the format, **"Case X: Y"**, where **X** is the case number and **Y** is the minimum time needed.

Constraint

- **1 ≤ T ≤ 4000**
- **1 ≤ N ≤ 2x10^5**
- **1 ≤ K ≤ N**
- **1 ≤ A[i] ≤ 10^6**
- **Sum of N over all test cases ≤ 2 x 10^6**

Sample Input	Output for Sample Input
3 5 4 10 13 6 7 9 5 5 10 9 8 7 12 5 3 7 5 3 11 2	Case 1: 2 Case 2: 1 Cass 3: 4

N.B.: Input file is huge. Please use faster I/O.

Problem C

Input: Standard Input
Output: Standard Output

Count the Attendance

In the world of competitive programming machine learning is rather an unknown and unnecessary topic. But in the professional world all the craze is about machine learning. Although the recent surge of cryptocurrency and blockchain based "Super Models" have everyone's attention, no one really understands them and some are very skeptic about the feasibility of these models given the available hardware.

In this problem you are going to solve a problem that will also give you an insight of the problems machine learning models solve. In particular there is a special group of problems called "unsupervised learning" where a machine learns to partition given data into groups.

For this problem you are given a rather interesting case of finding groups. You are going to try and find the different friend circles in a university! I know this last sentence is making your gossip heart cry out with excitement! Just think of all the possibilities if you could find out who is friend with who, who is dating who etc.

In universities people find freedom and with that freedom lies a specific behavior that you are going to exploit. In particular, students usually come to universities not to attend classes but to meet up with friends. There are some lone wolves who take class notes and who come to save everyone like Noah during exams with their metaphorical arc - class notes. But apart from these unique creatures there are several groups in a class who will usually be present either all together or none at all. Well maybe that's too extreme. Because this is not always the case. But it's usually the case most of the time.

You will be given an attendance sheet of **N** students of a particular class for **M** days. In order to find their various group structure and intricate social connections you will first need to find which students come to the class as a group. A group does not necessarily consist of all students present in a day and each possible subset of students can be a group. So you would like find out for each student the most likely group that they belong to. Each group has a specific weight to it. It is calculated by this function:

```
int weights[1 << N], seed;    // seed will be given as input, N is the number of students
void calc() {
      weights[0] = 1;
      for(int i = 1; i < (1 << N); i++)   weights[i] = (weights[i-1] * seed) % 10000;
}
```

The likelihood of a student belonging to a group is calculated by:

*Number of days every member of that group was present * Weight of that group*

For each student, you have to find all the groups with highest likelihood that the student is part of. For example, if a group is formed of students 2 and 3, then student 1 is not part of it, and it would be meaningless to calculate its likelihood for student 1. If a group is of the highest likelihood for one of its members, it may not be so for all other members as well. For example, person 3 may belong to a group consisting of person 2 and 3 whereas person 2 may belong to another group consisting of person 1 and 2.

For each day, you will be provided a **decimal mask**. A **decimal mask** is a decimal integer, whose binary representation will denote the attendance for a day. If the i-th bit (1-indexed) from the right is on, then we can say that the i-th student is present. For example, if students 3 and 4 are present in a day, then the binary representation has to be 1100. So the given **decimal mask** will be 12. Similarly, if students 1, 2 and 4 are present, the given **decimal mask** will be 11.

Input

The first line contains a number **T (1 ≤ T ≤ 10)** which represents the number of test cases. The first line of each test case consists of **3** integers, **N (3 ≤ N ≤ 20), M (0 < M ≤ 100000)** and **seed (0 < seed < 10000)** denoting the number of students in the class, the number of days of attendance you are given and the seed needed to calculate the group weights. Each of the next **M** lines contains an integer denoting the **decimal mask** (whose binary representation denotes the attendance of students of that day, mentioned earlier).

Output

For each test case print the case number in first line. After that, print one line for each student containing his/her maximum likelihood belonging to any group, followed by the **decimal mask** of that group. If multiple such group exists, you have to print **decimal masks** of all those groups sorted in ascending order separated by single space. Check sample for details.

Sample Input

```
2
5 6 2
4
11
10
17
19
23
3 3 2
5
6
5
```

Output for Sample Input

```
Case 1:
8608 23
8608 23
8608 23
2048 10 11
16608 16
Case 2:
64 5
64 6
64 5 6
```

Explanation of Sample I/O

In case 1, for students 1, 2, and 3, maximum likelihood is 8608, which can be obtained from group {1, 2, 3, 5}. For student 4, maximum likelihood is 2048, which can be obtained from groups {2, 4} and {1, 2, 4}. For student 5, maximum likelihood is 16608, which can be obtained from group {5}.

Problem D

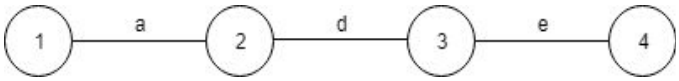
Input: Standard Input
Output: Standard Output

Save the World

Mr. MarZam lives in the techland. He works as a CID. He has a great reputation for his work. There is not a single investigation that he wasn't able to solve. There are **N** beautiful cities in the techland and they all are connected with **N-1** bidirectional roads.

A few days ago, a new gang of criminals has raised who are very jealous of seeing Techland's people in peace. So, they are planning to attack in some of the cities.

As Mr. MarZam has spies all over the country, he got the news that something wrong is going to happen. In order to protect people from these attacks, he wants one of his best spies, Mr. ShaVel, to infiltrate the gang and learn about their upcoming attack plans. But it comes out that it's not easy for Mr. ShaVel to communicate with Mr. MarZam when he is with the gang. So they made an alternate plan to communicate with each other. They named the cities with integers from **1** to **N** and labeled each of the roads with **lowercase** letters (**a** to **z**). Multiple roads can be labeled with the same letters. Mr. ShaVel will send a city name **U** and a sequence of letters **S** which means - there will be an attack on the city named **V** (**U** $\neq$ **V**) if the shortest path from **U** to **V** has the same letter sequence as **S**.



The sequence of letters from city 1 to city 3: *ad*
The sequence of letters from city 1 to city 4: *ade*

According to the plan, Mr. ShaVel joined the gang. After Mr. MarZam received the first message from his spy, he fell into a very critical situation. He figured out that there can be multiple cities which can have the same sequence of letters from city **U**. So, he needs to take protection in all those cities. But figuring out those cities may take too much time and delay the process to take preventive measures. As you guys are a great team and you are always ready to help people, Mr. MarZam has come to you for helping him with a piece of code to find out the cities. But to make your life a bit simple, Mr. MarZam only asked you to give him the total number of cities which may have an attack.

Input

The first line of the input will contain an integer **T (1 ≤ T ≤ 5)** which denotes the number of test cases.

Each test case starts with a line having two integers **N (1 ≤ N ≤ 10^5)** and **Q (1 ≤ Q ≤ 10^5)**. Each of the next **N - 1** lines will contain two integer $\mathbf{U_i}$**(1 ≤** $\mathbf{U_i}$ **≤ N),** $\mathbf{V_i}$ **(1 ≤** $\mathbf{V_i}$ **≤ N)** and a lowercase letter $\mathbf{C_i}$ which means there is a

bidirectional road between city $\mathbf{U_i}$ and $\mathbf{V_i}$ and the road is labeled with $\mathbf{C_i}$. Next, **Q** lines follow. Each of the lines will contain an integer $\mathbf{U_i}$ **(1 ≤ $\mathbf{U_i}$ ≤ N)** which is a city name and a non-empty sequence of lowercase letters $\mathbf{S_i}$ provided by Mr. ShaVel.

In a single input file, the sum of lengths for all the letter sequences will be less than or equal to $\mathbf{2 \times 10^6}$.

Output

For each test case, you should print the case number on the first line in the format: "**Case X:**" where **X** is the case number. Each of the next **Q** lines should contain the answer to Mr. ShaVel's messages. Please see the sample I/O for more clarification.

Sample Input

```
1
6 3
1 2 a
1 3 a
2 4 b
3 5 b
2 6 c
1 ab
2 b
1 ac
```

Output for Sample Input

```
Case 1:
2
1
1
```

N.B.: Input file is huge. Please use faster I/O.

Problem E

Input: Standard Input
Output: Standard Output

Alice & Her Cousins

Alice has a positive number of cousins. One day she wanted to distribute some chocolates among her cousins. All of her cousins had to get an equal number of chocolates because she didn't want them to know that she didn't love them equally. In order to fulfill her purpose, she went to a local grocery store and bought a box having **N** chocolates and **equally** distributed all the chocolates inside that box among all her cousins.

You are sure that if she had **Y** or **Z** number of cousins, then it wouldn't have been possible to distribute **N** chocolates equally among her cousins. Knowing this information, you have to tell if it is impossible for her to have **X** number of cousins.

Input

The first line of the input will contain an integer **T**, denoting the number of test cases. Then **T** lines will follow. Each line will have three **distinct** integers **X, Y, Z** separated by a space.

Constraints

- **$1 \leq T \leq 100$**
- **$2 \leq X, Y, Z \leq 10^9$**

Output

For each test case, print a line containing the case number and if it is impossible for her to have **X** number of cousins, then print "**Impossible**", otherwise print "**Possible**". Look at the sample input/output for more details.

Sample Input	Output for Sample Input
2 5 3 7 16 4 8	Case 1: Possible Case 2: Impossible

Problem F

Input: Standard Input
Output: Standard Output

Building Blocks

Jacob Builders is the largest builder in the city of Rozenthal. The city has become a hub of manufacturing, and this has created a huge problem for the residents. To control the chaos the city has limited the different types of blocks that can be manufactured at any time. Blocks are used to build pillars by stacking them one above another. Now Jacob Builders want to build a new skyscraper and need to know if they can construct some pillars with the new restriction. In order to find that out, they've come to you for help.

You are given an array of **N** integers, where each integer is the length of a pillar. Next, you are given the lengths of **K** blocks and a list of **Q** queries of the form **[L, R]**. For each query, you have to answer how many pillars of lengths between the **L** and **R** index of the array can be made using the **K** blocks. You can use any block any number of times.

For this problem you will given two integers A and B, and the lengths of the pillars can be generated using the formula:

$$Pillar[i] = (A \times i) + B \text{, where } 1 \leq i \leq N$$

Input

Input starts with a line containing the number of test cases, **T**. For each test case the first line will contain an integer, **N**, followed by a line containing two integers, **A** and **B**. After that you will be given a line with the integer **K** and **K** space separated integers on the next line. The next line will contain an integer **Q**. Each of the next **Q** lines will contain two integers **L** and **R**.

Output

For each test case, output the case number followed by results of the queries for that test case, each on a seperate line. See the sample output for more details.

Constraints:

- **$1 \leq T \leq 10$**
- **$1 \leq N \leq 10^{15}$**
- **$0 \leq A, B \leq 10^{15}$**
- **$1 \leq K, Q \leq 50$**
- **$1 \leq L, R \leq N$**
- **1 ≤ The length of any pillar ≤ 10^{15}**
- **1 ≤ The length of any block ≤ 10^{5}**

Sample Input

```
2
100
1 0
2
2 3
3
1 10
10 50
50 100
100
3 0
2
5 7
3
2 8
7 50
35 80
```

Output for Sample Input

```
Case 1:
9
41
51
Case 2:
4
44
46
```

Problem G

Input: Standard Input
Output: Standard Output

Interesting Device

The scientists of Gigaland have developed some interesting devices. Those devices can solve very complex calculations by communicating with each other. The devices are placed in various points of 3D space. The communication between any pair of devices is always performed in a straight line connecting the coordinates of both devices. Communications don't face any difficulty if other devices lie on the straight line connecting a pair of devices.

The government of Gigaland has decided to use those devices for weather forecasting. To do that with best performance, scientists have calculated optimal coordinates for each device. As the devices are very sensitive, all of those are covered by a special veil which separates the devices from the remaining universe(!), so that the devices can communicate with each other without any external barrier. They covered the devices in such a way that, every line of communication remains within the volume enclosed by the veil.The material of the veil is very rare and costly, that's why the covering was done with veil of minimum surface area. It is confirmed that the volume of interior space enclosed by the veil is always greater than zero.

After completing the setup they have faced an interesting issue: an invisible infinite plane which has passed through the veil hampering the communication between some devices. They have decided to destroy the portion of invisible plane which is inside the veil. To do that they need to calculate the area of that particular part. Help them to calculate the area. It is confirmed that this area is nonzero

Input

Input starts with an integer **T** (**T ≤ 25**) denoting the number of test cases followed by a blank line.

First three lines contains three integers giving the **X, Y** and **Z** coordinates of three noncollinear distinct points defining the infinite plane. Next line contains an integer **N** (**4 ≤ N ≤ 200**), the number of devices. Each of the following **N** lines contains three integers giving the **X, Y** and **Z** coordinates of a device. Absolute value of all the coordinates of input set is not greater than **100.**

Consecutive test cases are separated by a blank line.

Output

For each test case, print a line in the format, **"Case X: Y"**, where **X** is the case number and **Y** is the area of the portion of infinite plane which is inside the veil. Absolute error less than 10^{-3} will be ignored.

Sample Input

```
2

0 0 5
5 5 5
0 5 5
8
0 0 0
10 0 0
0 10 0
10 10 0
0 0 10
10 0 10
0 10 10
10 10 10

0 0 0
0 5 0
10 8 10
8
0 0 0
10 0 0
0 10 0
10 10 0
0 0 10
10 0 10
0 10 10
10 10 10
```

Output for Sample Input

```
Case 1: 100.00000000
Case 2: 141.421356237
```

Problem H

Input: Standard Input
Output: Standard Output

Skill Matching

BAPSIT is one of the most popular companies in the tech industry right now. They have been producing the most anticipated softwares in the market for years. But due to some budget allocation related issues, they are going to fire some of their employees as soon as possible.

There are a total of **N** employees currently working at BAPSIT. Each of those individuals has got a particular skill where they are very specialized at, as they have been working here for a long time. The skill set can be represented as integers in the range from **1** to **M**. Some employees have already made friends with each other and this friendship network can be represented as a tree. **It is to be noted that, for any pair of employees u and v, if u is a friend of v, then v is also a friend of u in this friendship network.**

Now the company wants to fire the maximum number of employees but at the same time, they also want to maintain all of their projects very smoothly. So they want at least one employee from each particular skill to stay at the company. Besides, they also want to make sure that, those who will eventually stay in the company, must be able to communicate with each other. For any pair of employees u and v, they can communicate with each other if they are directly friends with each other or they are connected with each other through some other employees in the remaining friendship network.

For example, let's say that employee 1 and 2 are friends, employee 2 and 3 are friends and at the same time employee 3 and 4 are also friends. In this example, all the employees will be able to communicate with each other. But if we fire the employee 3 from the company then the employee 1 and 4 won't be able to communicate with each other anymore.

Now as a senior project manager of BAPSIT, you have to find the maximum number of employees BAPSIT can fire.

Input

The first line of the input contains a single integer **T**, which denotes the number of test cases. The first line of each test case contains two integers **N** and **M**. The next line of a test case contains an array **S** of **N** space separated integers where $\mathbf{S_i}$ represents the skill of the i-th employee. Each of the next **(N - 1)** lines contains two integers **u** and **v** which denotes that the employees **u** and **v** are friends with each other.

Note: It is guaranteed that, in the initial friendship network provided in the input, all the employees are able to communicate with each other and there exists at least one employee from each particular skill.

Constraints

- **$1 \leq T \leq 10$**
- **$1 \leq N \leq 1000$**
- **$1 \leq M \leq 10$**
- **$1 \leq S_i \leq M$**
- **$1 \leq u, v \leq N$**

Output

For each test case, output a single line in the format "**Case X: D**" without the quotes. Here, **X** is the case number and **D** is the maximum number of employees the company can fire.

Sample Input

```
1
10 4
1 2 4 2 1 4 1 2 4 3
1 2
1 3
1 4
3 5
3 6
3 7
6 8
6 9
9 10
```

Output for Sample Input

```
Case 1: 4
```

Explanation

The company can fire the employees **2, 4, 5** and **7**. The remaining employees cover all the required skills and they can also communicate with each other either directly or through some of their friends.

Problem I

Input: Standard Input
Output: Standard Output

Almost Pattern Matching

Two strings **S1** and **S2** are said to be *almost equal* if they are of the same length and either of the following is true:

1. The strings **S1** and **S2** are equal, i.e, they are exactly the same.
2. **S1** and **S2** has exactly one mismatch between them, i.e, there is exactly one index *i* such that **S1[i] ≠ S2[i]**

Some examples of *almost equal* strings:

1. *hello* and *hello*
2. *good* and *food*
3. *aaabbb* and *aaaabb*

However, the following below are **not** *almost equal* strings:

1. *hello* and *helloo*
2. *feel and good*
3. *abcde and abdce*

Given a text **S**, and **Q** queries where each query contains a pattern **P**, count for each query how many substrings are there in the text which are *almost equal* to the pattern **P**.

Input

The first line of the input contains an integer **T**, denoting the number of test cases. Then the description of **T** test cases follow. The first line of every test case will contain the text **S**. The next line contains an integer **Q**. Each of the next **Q** lines will contain a pattern **P**.

Constraints

- **1 ≤ T ≤ 8**
- **1 ≤ |S|, |P| ≤ 262144**
- **1 ≤ Q ≤ 262144**

Sum of characters in the pattern across all queries in a single test case will not exceed **262144**.

|S| and **|P|** will be a **power of two** always. This means that **$|S| = 2^Y$**, where **Y** is any non-negative integer satisfying the constraints above. Similarly, **$|P| = 2^Z$**, where **Z** is any non-negative integer satisfying the constraints.

S and **P** will only contain characters from the **first 16** characters of the lowercase English alphabet **(a-p)**.

Output

For each test case, first print the case number in one line, like *"**Case X:**"*, where ***X*** is the case number starting from **1**. Then for each query, output the number of substrings in the text which are *almost equal* to the pattern **P**.

Sample Input

```
2
abcdcabcdacbfefe
8
abcd
dabc
dc
p
ff
acbd
gg
efef
aaaababa
4
bbbb
aaaa
abcdcabcdacbfefe
aa
```

Output for Sample Input

```
Case 1:
2
1
5
16
4
1
0
1
Case 2:
0
3
0
7
```

N.B.: Input is huge, please use faster I/O methods.

icpc international collegiate programming contest

ICPC Asia West Regionals 2019

ICPC Asia West Continent

Final Contest

Official Problem Set

icpc.foundation

icpc global programming tools sponsor

15th January 2020

You get 28 Pages, 13 Problems & 300 Minutes

This page is left intentionally blank.

Problem A: Q-SQL

Ahmad and Nabi just learned about SQL (Structured Query Language) in their database course. Because they love programming, they want to develop a small DBMS (Database Management System) as a fun project. They want to create a program that can run a very simple SQL SELECT query against a CSV (Comma Separated Values) file. For the first version of their program, they want to implement the functionality that supports the following SQL syntax:

```
SELECT columns
WHERE expression;
--------------------------------------------------
columns: * | column[,column...]
expression: column="value" | (column="value") | column="value" {AND | OR} expression | (column="value" {AND | OR} expression)
```

Note:

- There is no FROM clause; it is not supported as we are only reading data from a single file.
- WHERE clause is **optional**. eg: SELECT *; is a valid input.
- Only EQUALITY, AND, OR keywords/operators are supported, but they can be nested.
- In WHERE clause, column value is **always** wrapped in "double quotations". There won't be any whitespaces inside double quotations for column values.
- The SELECT statement and Equality operation are both **case insensitive**.
- Expressions are evaluated from left-to-right, and precedence is specified by parentheses.

Input

There will be only one case per input file. The first few lines contains the SQL select query, always ending with a **semicolon (;)**. The length of the query including white spaces and the semicolon will never be more than **666** characters. The select query can contain any **valid ASCII characters** but will follow the syntax specified above.

The next line contains **L**, the number of lines in the CSV file, **2 ≤ L ≤ 100000.** Then **L** lines will follow representing the CSV file. The first line will contain column names which will all be unique. Each of the next **L - 1** lines will contain the CSV data.

Column name and column value will only contain alphanumeric characters **(a-z, A-Z, 0-9)**. Neither column name or value will contain any of the special keywords like SELECT, FROM, WHERE, AND, OR, etc.

1 ≤ num of column, len(column name), len(column value) **≤ 100**

Note that the input file can be huge, please use faster I/O methods. However it is guaranteed that no input file will exceed 18 MB.

Output

For each query, first output the column names. For select * queries, print all the column names of the CSV as given in the input, **preserving their order and case**. For queries where the columns to be selected are given, print the selected columns **preserving the order and case specified in the select query**. The

order and case of columns in the query can be different than that in the CSV file. Lastly output all rows of the CSV file matching the query. The rows should be printed **in the order given in the input**, that is if row_1 and row_3 match, row_1 should be printed first followed by row_3. The column values for each row will however follow the **order of the column names printed in the last step, but the case of them must be preserved as in the CSV file**.

Note:
The query may have extra spaces. Note the case insensitiveness of the keywords, names and values too:
SelEct a, B, c where ((((A = "A" AnD b= "2")));

Sample Input

```
select a,   D  where A="1" and
(     b="1" or d="1");
13
a,b,c,d
1,1,1,1
1,1,1,2
1,1,1,1
1,1,2,2
1,1,2,1
1,1,2,2
1,2,1,1
1,2,1,2
1,2,1,1
1,2,2,2
1,2,2,1
1,2,2,2
```

Output for Sample Input

```
a,D
1,1
1,2
1,1
1,2
1,1
1,2
1,1
1,1
1,1
```

Explanation:

Result	a	b	c	d
	1	1	1	1
	1	1	1	2
	1	1	1	1
	1	1	2	2
	1	1	2	1
	1	1	2	2
	1	2	1	1
	1	2	1	2
	1	2	1	1
	1	2	2	2
	1	2	2	1
	1	2	2	2
a="1" and (b="1" or d="1")				

Problem B: Auxiliary Prime

A number is said to be auxiliary prime if it can be changed into a prime number by changing at most one of its digits. Note that you are not allowed to change the leading digit of the number to zero.

Given an integer **N**, check whether the number is auxiliary prime or not?

Input

The first line of the input contains an integer **T ($1 \le T \le 10^6$)**, denoting the number of test cases. Then the **T** test cases follow. For each case, the only line of input contains an integer **N ($1 \le N \le 10^6$)**.

Output

For each test case, output in a single line "**yes**" or "**no**" (without quotes) depending on whether the number is auxiliary prime or not.

Sample Input	Output for Sample Input
4 17 1003 200 10030	yes yes no yes

Explanation

- In the first test case, 17 is already a prime. Thus, it's an auxiliary prime.
- In the second test case, you can change the third digit (from the left) to 9 and get the number 1093, which is a prime. Thus, 1003 is an auxiliary prime.
- In the third test case, you can't change 200 to a prime number by changing at most one of the digits. So it's not an auxiliary prime.

NB: Large dataset, use faster I/O.

This page is left intentionally blank.

Problem C: Currently at Top

The website Codeforces has a recent action window. This window stores n active blog posts.

Every minute, a user visits the website and :

- With probability $p = \frac{s}{t}$, he chooses one of the n blog posts from the window randomly, and comments on it. On doing so, this post comes on the top. So if the posts from top to bottom were $A_1, A_2, \ldots, A_n$ and he comments on A_i, the new order of posts becomes $A_i, A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n$.
- With probability $1-p$, he posts a new blog post which appears on the top and removes the bottom most post from the window. That is, if the window looked like $A_1, A_2, \ldots, A_n$ from top to bottom and he posts a blog post B, it would look like $B, A_1, A_2, \ldots, A_{n-1}$ after the post.

Find the expected number of minutes after which the post currently at top is kicked out of the window.

The probability p is given as a rational number $\frac{s}{t}$. The answer can be written as $\frac{A}{B}$, where A and B are positive coprime integers. Print the value of AB^{-1} modulo 10^9+7, where B^{-1} denotes the modular inverse of B modulo 10^9+7.

Input

First line will contain a single integer **C (C < 20)**, denoting the number of test cases. Each of the next lines contains three integers, **n (1 ≤ n ≤ 10^9), s , t (0 ≤ s < t < 10^9 + 7, t ≠ 0)** for each case.

Output

For each case, print the required expected value modulo **10^9+7** in a single line.

Sample Input	Output for Sample Input
2 1 1 2 4 127749536 137669261	2 217930292

Explanation

In sample input # 1, there is only one blog post. It will be kicked out when a new blog post appears. The probability of succeeding is $\frac{1}{2}$ in every minute. So, the expected number of minutes is 2.

In sample input #2, if the current posts are $1, 2, 3, 4$ in order from top to bottom, then the window after one minute looks like:

- 1, 2, 3, 4 with probability $\frac{p}{n}$ (comment on 1)
- 2, 1, 3, 4 with probability $\frac{p}{n}$ (comment on 2)
- 3, 1, 2, 4 with probability $\frac{p}{n}$ (comment on 3)
- 4 ,1, 2, 3 with probability $\frac{p}{n}$ (comment on 4)
- 5, 1, 2, 3 with probability $1-p$ (new blog post 5)

We have to find the expected amount of time after which 1 is kicked out of the window. For example, one possibility is: (1, 2, 3, 4) -> (3, 1, 2, 4) -> (5, 3, 1, 2) -> (1, 5, 3, 2) -> (6, 1, 5, 3) -> (3, 6, 1, 5) -> (5, 3, 6, 1) -> (7, 5, 3, 6) taking 7 minutes.

This page is left intentionally blank.

Problem D: Shortest Path with Discounts

Townsville city contains **N** junctions. There are **M** directed roads which connect the junctions. Alice, a computer programmer, lives at junction **S**. She travels to her workplace, which is at junction **T**. Whenever Alice uses a road, she needs to pay some cost in rubles which is fixed for each road. Her company has given her **K** discount coupons. The i-th coupon provides a discount of $\mathbf{d_i}$ rubles. Each discount can be used at most once and for at most one road. Help Alice find the cheapest route from junction **S** to junction **T**.

Note:
- A road can be used by Alice at most once in her journey.
- There is at most 1 directed road from one junction to another.

Input

The first line of the input contains **C (1 ≤ C ≤ 20)**, number of test cases to follow.
The first line of each test case contains three space-separated integers **N (1≤ N ≤ 300), M (1 ≤ M ≤ 10^4)** and **K (1 ≤ K ≤ 10^4)**, representing the number of junctions, number of roads and number of discount coupons respectively. The second line contains two space-separated integers, **S** and **T (1 ≤ S, T ≤ N)**. The third line contains **K** space-separated integers, where the i-th integer represents $\mathbf{d_i}$ **(1 ≤ d_i ≤ 1000)**, the i-th discount coupon. Each of the next **M** lines contain three space-separated integers, **u, v (1 ≤ u, v ≤ N), w (1000 ≤ w ≤ 10^9)**, which represents a directed road from **u** to **v** with cost **w**.

Output

For each test case, print a single line containing an integer, which is the minimum cost that Alice needs to pay to go from **S** to **T**. If no path exists from **S** to **T**, print **−1**.

Sample Input	Output for Sample Input
2 3 4 1 1 3 10 1 2 1010 2 3 2000 1 3 1500 1 1 1100 3 1 1 1 3 10 1 2 1000	1490 -1

Explanation

For the first test case, one possible path is 1−>2−>3 and the discount coupon can be applied on the road (1,2) to get a cost of (1010−10)+2000=3000 rubles. However, the optimal path would be to take the road (1,3) and use the discount coupon there and pay 1500−10=1490 rubles.

This page is left intentionally blank.

Problem E: Make GCD

The gcd of a non-empty array is defined as the greatest common divisor of the values in the array. Define the beauty of an array as the sum of gcd of all of its subarrays. For example, the beauty of array [4, 6, 3] is 4 + 6 + 3 + gcd(4, 6) + gcd(6, 3) + gcd(4, 6, 3) = 4 + 6 + 3 + 2 + 3 + 1 = 19.

Given an integer **X**, find a non empty array of length less than or equal to 10^5 with the values in the range of **[1, 10^5]** such that the beauty of the array is equal to **X**, or claim that such an array doesn't exist.

Input

First line of the input file contains the number of test cases, **T (1 ≤ T ≤ 30)**. Following T line contains a single integer, **X (1 ≤ X ≤ 10^{13})**.

Output

For each test case, if there is no non empty array of length less than or equal to 10^5 with the values in the range **[1, 10^5]** which has a beauty value of **X**, print NO on a single line. Else, find such an array, say $A_1, A_2, ..., A_n$:

- Print YES on the first line.
- Print **n** on the second line.
- Print the space separated array $A_1, A_2, ..., A_n$ on the third line.

If there are more than one possible valid outputs, output any.

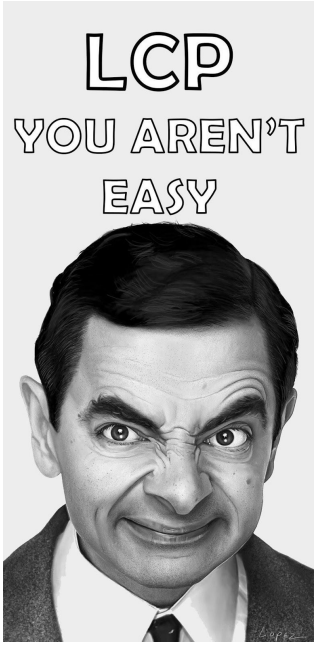
Sample Input	Output for Sample Input
2 19 19	YES 3 4 6 3 YES 1 19

This page is left intentionally blank.

Problem F: Finding LCP is not always easy

Mr. Bean is studying on string-processing algorithms. Recently he learnt about the term: Longest common prefix which is called LCP in short form. From one website, he got to know that LCP plays an important role in different string related data structures. It helps to easily find out which string can have less rank lexicographically. Sorting a list of string also depends on the LCP of two strings.

Mr. Bean's lovely teddy bear Beanie is really upset for the last few days. Because Mr. Bean wasn't giving enough time to Beanie as he was very busy with learning about LCP. Being angry, Beanie then asked Mr. Bean to solve a problem (though Beanie has no idea how to solve it).



Initially, Mr. Bean will be given a string **S** containing lowercase letters. You will have to perform the following operations on it:

- Insert a string of length **M** into **S** after the position **P** where each character of this new string will be **X**.
- Delete **N** characters from **S** starting from the position **P**. At any stage, the string length will be always greater than **N**.
- Reverse a substring of **S** where the substring starts from position **P1** to **P2**.
- Calculate the LCP of two substrings of **S**, one substring will start from position **P1** to **P2**, and another will start from **Q1** to **Q2**. These two substrings may overlap.

Mr. Bean is trying to figure out a solution to the problem for a long time. But he can't find any. Suddenly one idea came to his mind. There should be one or more people in his fan base who are very talented programmers. From a source, he got to know that you guys are waiting for problems so that you can solve. So, he is asking you for help. Can you help him solve the problem?

Input

The first line of the input contains a single integer **T** denoting the number of test cases.
Each case starts with a line containing the string **S**. Next line contains a single integer **Q** describing the number of operations need to perform on the string. Each of the next **Q** lines will contain any one type of the operations in the following format:

1 M P X: The insert operation
2 N P: The delete operation
3 P1 P2: The reverse operation
4 P1 P2 Q1 Q2: The calculation of LCP.

N.B.: Input file is huge. Please use faster I/O.

Constraints

- **1 ≤ T ≤ 5**
- **1 ≤ |S| ≤ 50,000**
- **1 ≤ Q, M, N ≤ 50,000**
- **1 ≤ P ≤ Length of S at that time**
- **1 ≤ P1 ≤ P2 ≤ Length of S at that time**
- **1 ≤ Q1 ≤ Q2 ≤ Length of S at that time**
- **The length of the string will never cross the limit of 10^6 at any stage**

Output

For each test case, the first line should contain the case number in the format "**Case Z:**" (without quotes) where **Z** is the case number. After that for operation type of **4**, you need to print the answer.

Sample Input	Output for Sample Input
1 abcde 5 1 1 4 a 4 1 2 5 6 1 1 4 b 3 5 6 4 1 2 5 6	Case 1: 1 2

Problem G: Fire in the Grid

FG is a board game played on **N** by **M** board (**N** rows and **M** columns). The left uppermost cell is numbered as $(1, 1)$ and the right lowermost cell is numbered as (N, M). The rules of the game is as followed.

1) Some of the cells contain bombs. These bombs are of two colors: Red and Green.
2) If a red bomb is fired, it fires all the bombs in its horizontal path. That is, a red bomb at cell (x,y) explodes all the bombs that belongs to that row **x**.
3) If a green bomb is fired at cell (x, y), it fires all the bombs which has column value **y**.

In the given figure you can see a 6 x 6 grid.
It has two red bombs at cell (2,1), (4,3) and two green bombs at cell (2,5) and (4,1).
There can be a series of bombing depending on the bomb we choose to fire.

Fire bomb at cell (2,1): It will fire bomb at (2,5). So, in total two bombs will be fired.
Fire bomb at cell (2,5): It will not fire anything else. So, only one bomb will be fired.
Fire bomb at cell (4,1): It will fire bomb at (2,1) and bomb at (2,1) will fire bomb at (2,5). So, in total three bombs will be fired.
Fire bomb at cell (4,3): All the four bombs will be fired.

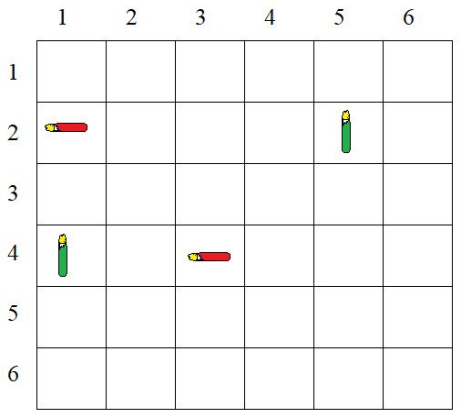


Figure: FG Game Board

In this problem, you are given the current state of the game. You have to decide the maximum number of bombs that can be fired by a single move.

Input

Input begins with an integer **T (1 ≤ T ≤ 10)** denoting the number of test cases. Each of the test cases starts with three integers **N, M** and **B (1 ≤ N, M, B ≤ 40000)** denoting the number of rows, columns and bombs. Each of the next **B** lines contains three integers **x, y** and **C** which denote there is a bomb of color **C** at **x-th** row and **y-th** column. **C = 0** means its a red bomb and **C = 1** means it is a green bomb.

Output

For each test case print, the case number followed by the maximum number of bombs that can be fired by a single move.

Sample Input	Output for Sample Input
1 5 5 4 2 1 0 2 5 1 4 1 1 4 3 0	Case 1: 4

NB: Large dataset, use faster I/O.

This page is left intentionally blank.

Problem H: Sarfaranga Rally

Sarfaranga is a desert in the northern areas of Pakistan. It is a major town in the Baltistan region which abounds in natural beauty. Located at the foothills of the mighty Karakoram Range, it is often the last town where mountaineers gather to give final touches to their K2 (the second highest peak in the world) expedition. Sarfaranga is located just outside of Skardu and every year a jeep rally event is organized there to see the best drivers and cars that can cross the desert. Many drivers from far and wide participate in this rally both for passion and for the prestige.

The rally has been on-going for several years with familiar faces and, many times, familiar winners. This year its organizers thought of a way to "spice things up" a bit. They thought that in addition to seeing who's got the best car and driving skills, they will also see who the smartest driver is. To achieve that, they came up with an interesting plan. The rally has been divided into **N** stages each consisting of **M** blocks like a 2D grid fashion. Stages are numbered from **1 to N** (top to bottom) and blocks of each stage are numbered from **1 to M** (left to right). Each block has an entry/exit gate to an adjacent block such that a driver can only move to a top, bottom, left, or right block from his/her current block position. Each block has a penalty to cross through that block. All drivers must start from the **1st block of the 1st stage** and end up in the **Mth block of Nth stage**. Note that, drivers can not move out of the grid as it will cause immediate disqualification.

You are a rally driver in the Sarfaranga desert rally. Just before the start of the race, each driver is provided with a map of the rally stage that includes the penalty associated with each block. You job is to quickly figure out the path that will result in the least penalty to win the race.

Input

The first line in the input file is an integer **T (1 ≤ T ≤ 50)** denoting the number of test cases. Each test case starts with two integers **N and M (1 ≤ N, M ≤ 20)**. Next line of the test case contains stage-wise entry of the penalty of each block **(0 ≤ Penalty of each block ≤ 200)**.

Output

For each test case, print a single line with the case number followed by the minimal penalty incurred if we follow the optimal path. See samples below for more clarification.

Sample Input	Output for Sample Input
2 3 3 1 2 3 5 4 3 1 0 5 4 4 1 1 1 0 4 5 6 0 4 4 4 0 5 7 8 1	Case 1: Penalty = 12 Case 2: Penalty = 4

This page is left intentionally blank.

Problem I: Subarray Optimization

You are given a sequence **P** of **n** lowercase English letters.
For any subarray (i, j), i.e. $P_i, P_{i+1}, \ldots, P_{j-1}, P_j$, we define:

Min(i,j) = minimum element of the subarray

Distinct(i,j) = number of distinct elements in the subarray

You need to calculate the following expression:

$$\sum_{i=1}^{n} \sum_{j=i}^{n} Min(i,j) \times Distinct(i,j)$$

Input

The first line of the input contains **t (1 ≤ t ≤ 5)** - the number of test cases.
Each test case is described as follows:

- The first line contains **n (1 ≤ n ≤ 10^5)** - the length of the sequence **P**.
- The second line contains **n** space separated integers, **P_i (1 ≤ P_i ≤ 10^5)** - denoting the sequence.

Output

For each test case, output in a single line the value of the expression in the statement.
It can be guaranteed that the answer will always fit in a 64-bit signed integer.

Sample Input	Output for Sample Input
3 3 1 2 3 3 1 2 1 3 1 2 2	15 10 11

This page is left intentionally blank.

Problem J: How Many Solutions?

Given the value of integer **N** how many solutions does the following equation have?

$$\frac{1}{x} + \frac{1}{y} = \frac{1}{N}$$

If **x** and **y** are integers there is only a finite number of solutions but if **x** and **y** are real numbers then there can be an infinite number of solutions. What if **x** and **y** are floating-point numbers with limited size, e.g. **x** and **y** are floating point numbers with **d** digits after the decimal points, how many different solutions will be there?

Input

Input file contains at most **2000** lines of input. Each line contains two integers **N (0 < N ≤ 10000000000)** and **d (0 ≤ d ≤ 1000)**, here **d** means that there can be maximum **d** digits after the decimal point. Input is terminated by a line containing two zeros. This line should not be processed.

Output

For each line of input, produce one line of output which contains an integer **T**. This line contains the number of different solutions the equation has for the given value of **N** and **d**. As the value of **T** can be very large so output **T** modulo **1000007**.

Sample Input	Output for Sample Input
23 10 10 2 0 0	2645 97

This page is left intentionally blank.

Problem K: Base Stations

T-net, a cellular network company, has established **n** base stations at **n** distinct positions, and assigned a frequency to each base station. Each base station covers clients within a certain distance. In general, the coverage area of base stations may overlap. This may lead to interference of the signals for a client who is in the coverage area of more than one base station. Thus, for a client within the coverage area of at least one base station, there must be a base station with a unique frequency among base stations covering the client. So, the client can connect to the network without interference. A point **p** is called a covered place if it is covered by at least one base station. A covered place **p** is called conflict-free if among base stations covering **p**, there is a base station with a unique frequency (i.e. there exists a base station whose frequency is different from the frequency of the other base stations covering **p**). Indeed, a client standing at a conflict-free covered place can connect to the network without interference. Two covered places **p** and **q** are called equivalent if all points on the segment **pq** (including **p** and **q**) are covered by the same subset of base stations. Otherwise, they are called non-equivalent. Now, T-net is wondering maximum how many covered places exist in the coverage area of its network that are not conflict-free and are pairwise non-equivalent.

For your ease, assume that the world is one dimensional. So, the coverage area of each base station can be modeled by a closed interval **[a, b]**, where **a** and **b** are two distinct numbers. Also, assume that the frequency assigned to each base station is a positive integer.

Input

There will be multiple test cases. You must take input until end of file. For each case, the first line of the input contains an integer **n** denoting the number of base stations **($1 \le n \le 10^5$)**. Each of the next **n** lines describes a base station by three positive integers **a**, **b**, and **f**, where **[a, b]** denotes the coverage area of the base station, and **f** denotes its frequency **($1 \le a < b \le 10^5$** and **$1 \le f \le 10^5$)**. **I/O file is huge, please use faster I/O.**

Output

For each case, print the number of pairwise non-equivalent covered places that are not conflict-free in a single line.

Sample Input

```
3
1 5 1
2 7 1
3 4 2
4
1 2 1
1 2 1
2 3 2
2 3 2
4
1 2 1
2 3 1
3 4 1
1 4 1
```

Output for Sample Input

```
2
3
5
```

Problem L: Table

Sarina has an **m x n** table whose cells are either 0 or 1. She wants to apply a series of operations on the table in a given order. Each operation involves toggling a subset of cells, where toggling a cell means changing it from 0 to 1, or vice versa. The operations are of the following types:

- $r\,x$: toggle all cells in every row whose number of 1 cells modulo 2 equals **x**.
- $c\,x$: toggle all cells in every column whose number of 1 cells modulo 2 equals **x**.

Kindly help Sarina to apply all the given operations and output the final table.

Input

There will be multiple test cases. You must take input until end of file. For each case, the first line of the input contains three positive integers **m**, **n**, and **q**, where **m** and **n** denote the number of rows and columns of the table, respectively **($1 \le m * n \le 10^5$)**, and **q** is the number of operations **($1 \le q \le 10^5$)**. Each of the next **m** lines describes a row of the table as a string of 0s and 1s of length **n**. The next **q** lines describe the list of operations that must be applied to the table in the given order. In all operations, **x** is either 0 or 1.

Output

For each case, print the final table after applying all operations.

Sample Input	Output for Sample Input
3 3 2 110 001 111 r 0 c 1 3 3 1 110 001 111 r 0	110 110 000 001 001 111

NB: Large dataset, use faster I/O.

This page is left intentionally blank.

Problem M: Coloring Graph

You are given a connected simple undirected graph with **n** vertices numbered **1** through **n**, and **m** edges. You have to color the graph in two colors numbered **1** and **2**, such that each vertex has exactly one neighbor whose color differs from its own, or claim that there doesn't exist such a coloring. If there are multiple correct answers, you can print any of them.

Input

First line will contain **T (1 ≤ T ≤ 100)**, number of test cases. Then the test cases follow.

The first line of each test case contains two space separated integers **n (1 ≤ n ≤ 1000)** and **m (n-1 ≤ m ≤ n+10)**, the number of vertices and the number of edges in the graph respectively.
- **i**-th of of the next **m** lines contain two integers $\mathbf{a}_i$ and $\mathbf{b}_i$ denoting an edge between vertices $\mathbf{a}_i$ and $\mathbf{b}_i$

You can assume that (i) the sum of **n** over all test cases doesn't exceed **15000** (ii) The graph described is a simple graph, i.e. there are no self loops and multiple edges.(iii) The graph described will be connected, i.e. for any pair of nodes, there exists a path between them.

Output

For each test case:
(i) If no valid coloring exists print **0** on a new line
(ii) If a valid coloring exists, print **1** on a new line and then print **n** space separated integers on another new line, where the **i**-th integer is either **1** or **2** denoting the color of node **i**.

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

This page is left intentionally blank.

ACM ICPC
Dhaka Regional Online Preliminary Contest
2018

5th October 2018

You get 13 Pages

10 Problems

A	Welcome **Input:** Standard Input **Output:** Standard Output	

ACM ICPC Dhaka Regional is the biggest programming competition in Bangladesh. Also the most anticipated one as well. Students from all the different universities storm their brains all year in preparation for this competition. You are now taking part in this competition, so a big congratulations to you.

Dhaka site is one of the biggest sites in ACM ICPC. This year almost 1500 teams are participating in the competition. So in celebration to that, we are going to give you an easy problem to start with.

You have to write a program, which will print the line "Welcome to ACM ICPC Dhaka Regional Online Preliminary Contest, 2018" (without quotes).

Note: you can't output anything other than the required output, and each line must end with a newline ('\n'). Take special care about spelling and case. If you alter any of those, you may not get accepted.

For your convenience, we are providing one sample program in C/C++ which prints "Bangladesh". You just have to change the code to your requirement.

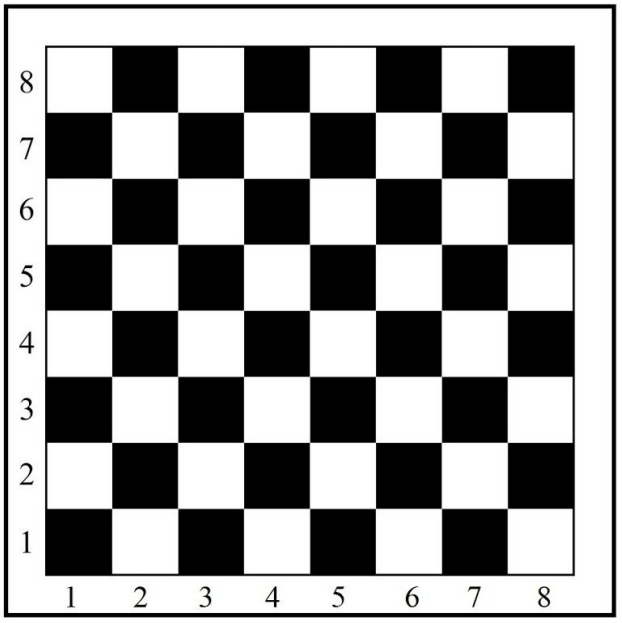
```
#include <stdio.h>
int main()
{
    printf("Bangladesh\n");
    return 0;
}
```

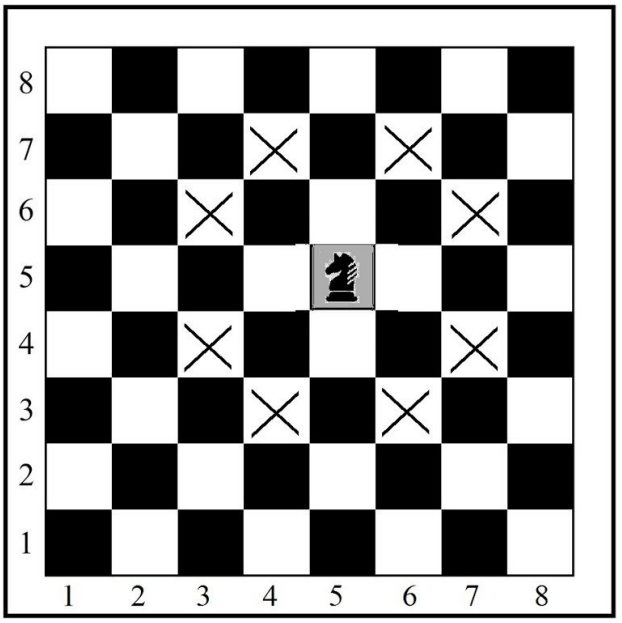
B Boring Chess

Input: Standard Input
Output: Standard Output

"Chess holds its master in its own bonds, shackling the mind and brain so that the inner freedom of the very strongest must suffer." – **Albert Einstein**

Chess is played on a chessboard, a square board divided into 64 squares (eight-by-eight) of alternating color. Knight is a special type of chess piece which moves in an interesting way. Its moves are like "L" shape.
A knight can move from cell (r_1, c_1) to cell (r_2, c_2) if and only if $(r_1 - r_2)^2 + (c_1 - c_2)^2 = 5$. Also, note that a chess piece cannot go out of the board.





In this problem, you are given the current position of a knight **(r, c)**. You have to find, the number of different cells where the knight can go in a single move from its current position.

Input

Input starts with an integer **T (1 ≤ T ≤ 64)** denoting the number of test cases. Following **T** lines will contain two integers **r, c** where **1 ≤ r, c ≤ 8**.

Output

For each test case, output the number of different cells the knight can go in a single move. For more clarity, see sample input/output.

Sample Input	Output for Sample Input
4 5 5 1 2 4 5 1 4	Case 1: 8 Case 2: 3 Case 3: 8 Case 4: 4

C	Odd is real **Input:** Standard Input **Output:** Standard Output	

Given a range **[L, R],** find the number of integers in that range which have an **odd number of odd divisors**. For example, 1 has only one divisor and it is odd, 9 has three divisors {1, 3, 9} and all of them are odd. Meanwhile, 18 has six divisors {1, 2, 3, 6, 9, 18} but three of them are odd. So 1, 9 and 18 have an odd number of odd divisors.

Input

Input will start with a positive integer **T ($T \leq 10^5$)** denoting the number of test cases. Each test case will have two positive integers **L, R ($1 \leq L \leq R \leq 10^{18}$)**, the range.

Output

For each test case, the first line will be the case number in the format **"Case t: x"** without the quotes. Here **t** is the case number starting from 1 and **x** is the number of integers that fall in the range **[L, R]** and have an odd number of odd divisors.

Sample Input	Output for Sample Input
3 1 3 5 10 10 15	Case 1: 2 Case 2: 2 Case 3: 0

D	Prime Friendly Numbers **Input:** Standard Input **Output:** Standard Output	

Given **N,** find the largest number **X,** not greater than **N,** such that **X** is **prime friendly**. A number is called **prime friendly** when it satisfies both of the following conditions:

1. The number itself is a prime.
2. All its digits in **base 10** are also primes. In other words, the number consists of only the digits **2, 3, 5, 7.**

Input

The first line contains an integer **T,** denoting the number of test cases. Each test case contains a single positive integer **N**.

Output

For each test case, output the case number followed by the largest number **X**, not greater **N**. Please refer to the sample input/output section for more clarity of the format.

Constraints:

$1 \leq T \leq 1000$

$2 \leq N \leq 10^{18}$

Sample Input	Output for Sample Input
5 10 100 1000 10000 100000	Case 1: 7 Case 2: 73 Case 3: 773 Case 4: 7757 Case 5: 77773

E	**The End** **Input:** Standard Input **Output:** Standard Output	

You are given a 2-dimensional grid consisting of **N** rows and **M** columns. Rows are numbered from 1 to **N** and columns are numbered from 1 to **M**. You are on the **(1, 1)** cell. You have to go to the **(N, M)** cell.

In each move, you can go to the next cell of the same row or next cell of the same column. For example, if you are on the **(x, y)** cell, then in the next move you can go to the cell **(x, y+1)** or **(x+1, y)**. You can't move outside the grid.

There are also **K** blocked cells on the grid. You can't go to any blocked cell.

Each time, before starting your move, you have **B** magical power. You can enter at most B blocked cell. For the next move after entering the blocked cell on **(x, y)**, you can go either **(x+1, y)** or **(x, y+1)**.

For Example, you are given 3x3 Grid. You have 2 blocks on **(1,2)** and **(1,3)**. You have 1 magical power.

Here are some ways

1. Using one magical power, you can go to the target by path **(1,1) (1,2), (2,2), (3,2), (3,3)**
2. Using zero magical power, you can go to the target by path **(1,1), (2,1), (3,1), (3,2), (3,3)**

But you cannot go to the target cell by path **(1,1), (1,2), (1,3), (2,3), (3,3)** because you need to enter 2 block cells but you have only one magical power.

You have to calculate how many ways you can go from **(1,1)** cell to the target cell **(N, M)**.

N.B.: Two ways are different if there is at least one different cell used in their path from the cell (1,1) to cell (N, M).

Input

Input starts with test case number **T ($1 \le T \le 10$).** For each test case, the first line contains two integers **N ($1 \le N \le 10^6$)** and **M ($1 \le M \le 10^6$)**, denoting the number of rows and columns of the grid. Next line contains two integers **K ($0 \le K \le 100$)** and **B ($0 \le B \le K$)**. Next, each of the K lines contains 2 integers denotes the row **R ($1 \le R \le N$)** and column **C ($1 \le C \le M$)** of the block. You may safely assume that no two blocks will be in the same position. There is no block on the start cell and target cell.

Output

For each test case, output the answer in a single line according to the problem. Since the answer can be very large, simply output the answer modulo **1000000007 ($10^9 + 7$)**.

Sample Input	Output for Sample Input
3 3 3 3 1 1 3 2 2 3 1 3 3 3 0 3 1 2 2 1 3 3 3 2 1 1 2 1 3	6 0 5

F	**Find the Substrings** **Input:** Standard Input **Output:** Standard Output	

You are given a string **S** and **Q** queries. On the **i**th query, you will be given two integers $\mathbf{N_i}$ and $\mathbf{M_i}$. Now for each query, you have to find such strings consisting of lowercase letters whose lengths are at least $\mathbf{N_i}$ and at most $\mathbf{M_i}$ and also those strings are not substrings of **S**.

For example:
You are given **S="abac"**. If a query contains $\mathbf{N_i=3}$ and $\mathbf{M_i=4}$ then according to our problem, **"aba"**, **"bac"** and **"abac"** are not valid as they are substrings of **S** while **"abc"** is a valid string as it is not a substring of **S** and also maintaining the constraints for the length. Now you have to find all other strings which are also valid for this string **S**.

To make the problem a bit simple, you don't have to print all the strings. You will just need to print the number of such strings which are valid. To make the output more simple, print the answer modulo **10^9+7**.

Input

The first line of the input file will be a single integer **T (1 ≤ T ≤ 5),** denoting the number of test cases.
Each test case contains two lines. The first line will contain the string **S (1 ≤ |S| ≤ 1000000)** of lowercase letters and the second line will contain an integer **Q (1 ≤ Q ≤ 100000)**. Each of the next **Q** lines will contain two integers $\mathbf{N_i}$ and $\mathbf{M_i}$ **($1 \le N_i \le M_i \le |S|$).**

Output

For each test case, the first line of the output should contain the case number in the format: **"Case X:",** where **X** is the test case number. Each of the next **Q** lines should contain the answer for the specific query modulo **1000000007** (**10^9+7)**.

Sample Input	Output for Sample Input
1 abcab 5 1 2 2 2 3 5 1 1 1 5	Case 1: 696 673 12355922 23 12356618

N.B. Dataset is large. Use faster I/O methods.

G	Subset with GCD K **Input:** Standard Input **Output:** Standard Output	

Given a set of **N** positive distinct integers and a query value **K**, find if there exists a subset of the integers whose greatest common division equals to **K**.

For example, if the given set is **{1, 6, 2, 9, 8}**, then the answer for the following values of **K** are:

Value of K	Existence of Subset	Possible Subset
1	Yes	{1}
2	Yes	{2,6}
3	Yes	{6,9}
4	No	-

Input

The first line contains a single integer **N (1 ≤ N ≤ 100000)**. The next line contains **N** positive distinct integers separated by whitespace. These **N** integers represent the set as mentioned above.

The third line contains a single integer **Q (1 ≤ Q ≤ 1000)**. The next line contains **Q** distinct positive integers, where each integer is a query **K**.

All values given will be less than or equal to 10^9. See sample I/O for more details.

Output

For each query, output in a single line the character **"Y"** if a subset exists whose GCD equals to **K** or **"N"** if not, without the quotes.

See sample I/O for more details.

Sample Input	Output for Sample Input
5 1 6 2 9 8 4 1 2 3 4	Y Y Y N

H Colorful Balls

Input: Standard Input
Output: Standard Output

Alice has **N** balls arranged in a single line. The balls are either red(**R**), blue(**B**), green(**G**) or white(**W**). They can be represented by a string **S**. Every character of the string is either **R, B, G or W**.

In the beginning, also there are no two balls of the same color side by side (except white ball). For example GGWWB is not a valid string because there are two green balls together. But GWWB is a valid string as there are no two balls of same color side by side except white balls.

Alice needs to paint all the white balls in one of the other three colors in a way that there are no two balls of the same color side by side.

How many ways Alice can paint the balls? Print the solution modulo **1000000007** (**$10^9 + 7$**).

Input

The first line contains the number of test cases **T (1≤T ≤1000)**. In each line of the test cases, there will be a string **S** of length **N (1≤N ≤100000)**.

The total number of character in the input file will be less than **$5 * 10^6$**.

Output

For each test case, print the case number and the answer to the problem.

Sample Input	Output for Sample Input
2 WWG GWGWB	Case 1: 4 Case 2: 2

Explanation for case 1: The four valid ways to color the balls are RBG, BRG, GRG, GBG.

N.B. Dataset is large. Use faster I/O methods.

I	Vugol Search **Input:** Standard Input **Output:** Standard Output	

"**Did You Mean**…" is a search engine function that scans for potential spelling or grammatical errors in user queries and recommends alternative keywords, similar to the auto-correction feature found in mobile messaging services. While the feature is designed to assist users in refining their search results, it has been frequently exploited by **Vugol search** users for comedic purposes. In Vugol search engine, people can search exactly one word at a time.

There are N words in Vugol's database. Each word has its **Smartness** value. If someone does search in the search engine with a word then Vugol search engine calculates the score of each N words with respect to the searched word and suggest the word which has the maximum score with respect to searched word. If the searched word is A, the score is calculated with respect to word B as,

score(A, B) = LCP(A, B) + IsAnagram(A, B) * Smartness value of B
Where:

- LCP(A, B) = length of the longest common prefix of two string A and B.
- IsAnagram(A, B) = If A is anagram of B then 1 else 0.

One or more words from Vugol's database can have the same maximum score with respect to searched word.

You are given Q most searched words in the Vugol search and you already know Vugol's database have N words. For each searched word you have to find the score of the word that Vugol suggests.

Note: An **anagram** of a string is another string that contains the same characters, only the order of characters can be different. For example, "abcd" and "dabc" are anagram of each other.

Input

The first line contains an integer **T,** denoting the number of test cases.
For each test case:

- The first line contains an integer **N** denoting the number of words.
- The second line contains an array of **N**-space separated positive integers denoting Smartness value of words.
- Next **N** lines contain a word.
- Next line contains an integer number **Q**, denoting the number of queries.
- Next, **Q** lines contain a query word.

Constraints

- **1 ≤ T ≤ 10**
- **1 ≤ N, Q**, Smartness value **≤ 10^5**
- A word consists of **only lowercase** English alphabets.
- For **each test case**, the sum of all words' length is within **2 * 10^5**, including query words.

Output

For each query, print the maximum score of a word that Vugol suggests.

Sample Input	Output for Sample Input
1 4 5 2 4 10 google oooogp googoo goloeg 5 google lgooeg glooeg poops goopoo	12 10 11 0 3

N.B. Dataset is large. Use faster I/O methods.

J Yet Another Longest Path Problem

Input: Standard Input
Output: Standard Output

You will be given a graph of **N** nodes connected by **N-1** edges where all the nodes are directly or indirectly connected together. Now we want to modify the graph by making the edges directed. You can visit a node from another one following a path formed by these directed edges. While transforming the graph into a directed one, we also want to minimize the length of the longest possible path in the graph.

You have to determine the direction of each edge to achieve this goal.

Input

There will be several test cases, **T (1 ≤ T ≤ 20)**. For each test case, the first line will contain **N (2 ≤ N ≤ 100000)** denoting the number of nodes. Each of the following **N-1** lines will contain two integers **u, v (1 ≤ u, v ≤ N)** denoting an undirected edge between nodes **u** and **v**.

Output

For each test case, print a line in the format "**Case X:**" where **X** is the integer denoting the test case number starting from 1, then print **N-1** lines, each containing two integers **u** and **v** denoting that there is a directed edge from node **u** to **v** in the graph after the transformation as described above. If there are more than one solutions, print **any** of them, and print in **any order** you want.

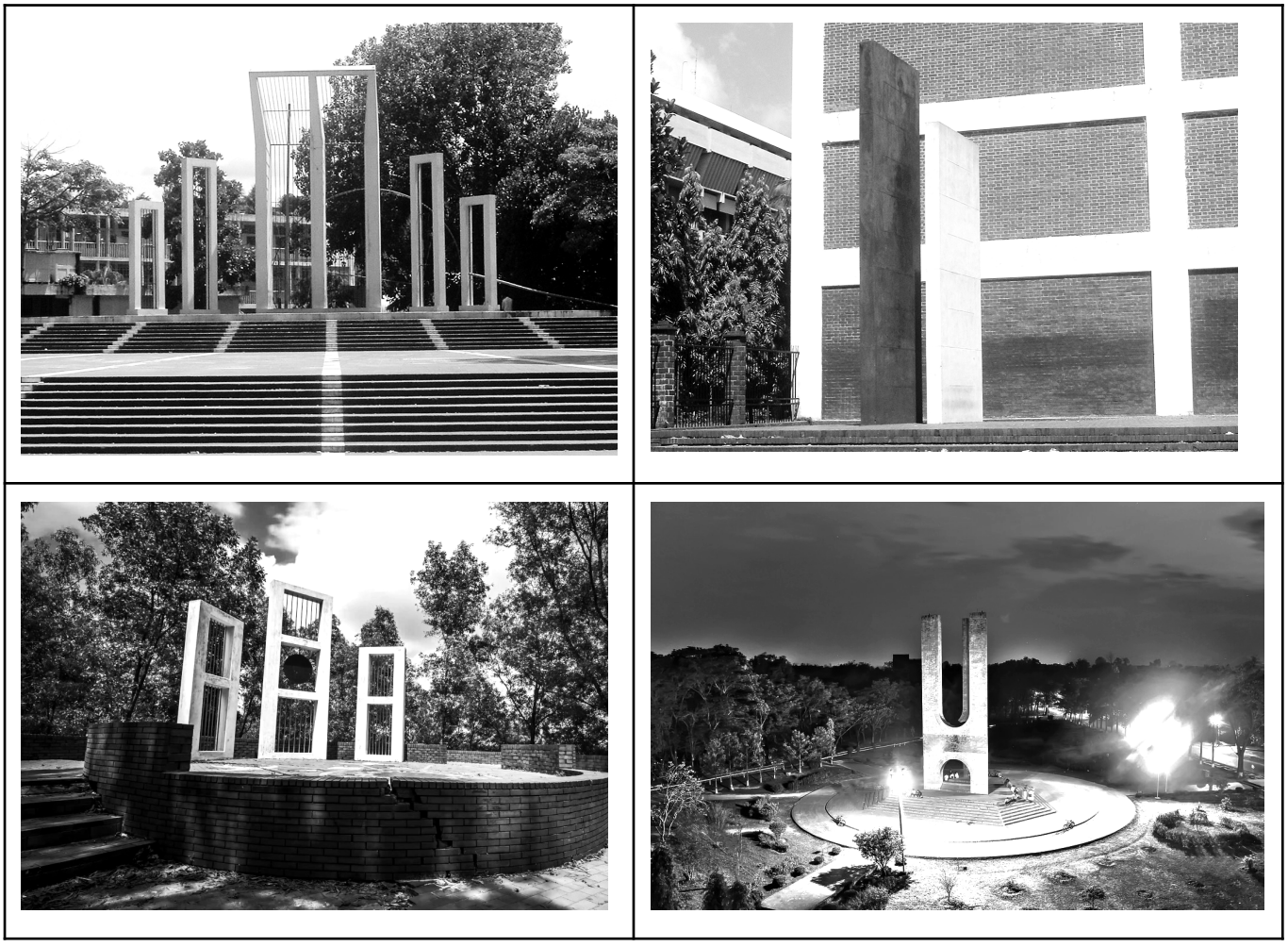
Sample Input	Output for Sample Input
2 3 1 2 2 3 4 2 1 1 3 4 1	Case 1: 1 2 3 2 Case 2: 1 2 1 3 1 4

This is a special judge problem.
N.B. Dataset is large. Use faster I/O methods.

icpc.foundation
icpc International Collegiate Programming Contest
Daffodil International University
icpc global programming tools sponsor

Official Problemset
ICPC Dhaka Regional 2018

```
do
{
    printf("Do not touch anything\n");
} while("Not asked to touch");
```



You will get:
10 Problems, 16 Pages, 300 Minutes

Problem A

Input: Standard Input
Output: Standard Output

Average of Combination

Given **N** numbers (1, 2, 3 … N), you can take **R** different numbers in several ways. For example, if **N = 4** and **R = 3**, there are 4 possible ways:

- 1, 2, 3
- 1, 2, 4
- 1, 3, 4
- 2, 3, 4

If we fix **N**, and vary **R**, we will get many different combinations. In total, for a particular **N**, there can be $2^N - 1$ different combinations. Let's define two statistics **S** and **A** for a combination, where **S** is the sum and **A** is the average of the numbers in the combination. For your convenience, these values (R, S and A) are shown below for all the combinations of **N = 4**:

R	Combinations	Sum (S)	Average (A)
1	1	1	1
	2	2	2
	3	3	3
	4	4	4
2	1, 2	3	1.5
	1, 3	4	2
	1, 4	5	2.5
	2, 3	5	2.5
	2, 4	6	3
	3, 4	7	3.5
3	1, 2, 3	6	2
	1, 2, 4	7	2.33…
	1, 3, 4	8	2.66…
	2, 3, 4	9	3
4	1, 2, 3, 4	10	2.5

You can see that, the average of different combinations may vary. We are interested in ordering the combinations according to their average (non-increasing order). If two combinations have the same average, then the one with fewer numbers will come earlier in order. If two combinations have the same average and the same number of elements then the lexicographically (considering the combinations as sorted sequence of integers) smaller one will come earlier in order. So for **N = 4**, the combinations in order are:

1. 4
2. 3, 4
3. 3
4. 2, 4
5. 2, 3, 4
6. 1, 3, 4
7. 1, 4
8. 2, 3
9. 1, 2, 3, 4
10. 1, 2, 4
11. 2
12. 1, 3
13. 1, 2, 3
14. 1, 2
15. 1

For this problem, given **N** and **K**, you have to print the **K^{th}** (1-based) combination of the numbers in the range from **1** to **N** in the order mentioned above.

Input

First line will contain number of test cases, **T (1 ≤ T ≤ 100)**. Each case will contain two integers **N (1 ≤ N ≤ 100000)** and **K (1 ≤ K ≤ min (2 × 10^7, 2^N-1))**.

Output

Print case number and then the required combination in each line, separated by spaces. Please see the sample for clarification.

Sample Input	Output for Sample Input
4 4 7 4 15 4 1 10 30	Case 1: 1 4 Case 2: 1 Case 3: 4 Case 4: 4 7 8 9 10

Problem B

Input: Standard Input
Output: Standard Output

Counting Inversion

The number system we are used to is called the decimal number system. The digits of decimal number system are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

$$(123)_{10} = 1 \times 100 + 2 \times 10 + 3 \times 1 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

The leftmost digit is called the most significant digit and the rightmost one is called the least significant digit. Let, **K** be a decimal integer represented as $\mathbf{d_n \ldots d_2 d_1 d_0}$, where, $\mathbf{d_n}$ ($\mathbf{d_n > 0}$) is the most significant digit (the leftmost digit) and $\mathbf{d_0}$ is the least significant digit (the rightmost digit).

Any two digits of **K**, $\mathbf{d_i}$ and $\mathbf{d_j}$ form an inversion **if and only if,** $\mathbf{d_i > d_j}$ where $\mathbf{i < j}$.

We define the number of digit inversions of **K** as **DI(K)**. For example, if **K = 123,** then, $\mathbf{d_2 = 1, d_1 = 2, d_0 = 3}$ and **DI(K) = DI(123) = 3** (as $\mathbf{d_0 > d_1}$, $\mathbf{d_1 > d_2}$ and $\mathbf{d_0 > d_2}$). If **K = 253,** then $\mathbf{d_2 = 2, d_1 = 5, d_0 = 3,}$ then **DI(K) = DI(253) = 2** (as $\mathbf{d_0 > d_2}$ and $\mathbf{d_1 > d_2}$). Similarly, **DI(5) = 0 , DI(321) = 0, DI(491383) = 6**.
In this problem, you are given two integers **x** and **y** in the decimal number system.
You have to calculate $\sum_{K=x}^{y} DI(K)$.

Input

Input starts with an integer **T($1 \le T \le 50000$)** denoting the number of test cases.
Following **T** lines each contains two integers **x** and **y** (without leading zeros) where **($1 \le x \le y \le 10^{14}$)**. The dataset is huge, so use faster I/O methods.

Output

For each test case, the output should contain the case number in the format: "**Case T:** ", where **T** is the test case number followed by the desired answer in a single line. Please see the sample for clarification.

Sample Input

```
5
1 9
1 100
50 60
23 2343
345 99373
```

Output for Sample Input

```
Case 1: 0
Case 2: 36
Case 3: 4
Case 4: 6083
Case 5: 410008
```

Problem C

Input: Standard Input
Output: Standard Output

Divisors of the Divisors of an Integer

The function **d(n)** denotes the number of positive divisors of an integer **n**. For example **d(24) = 8,** because there are **8** divisors of **24** and they are **1, 2, 3, 4, 6, 8, 12** and **24**. The function **sndd(n)** is a new function defined for this problem. This denotes "The summation of number of divisors of the divisors" of an integer n. For example,

sndd(24) = d(1) + d(2) + d(3) + d(4) + d(6) + d(8) + d(12) + d(24) = 1 + 2 + 2 + 3 + 4 + 4 + 6 + 8 = 30.

Given the value of **n,** you will have to find **sndd(n!)**, here **n!** means factorial of **n**. So **n! = 1 × 2 × 3 × ... × n**.

Input

The input contains at most **1000** lines of test cases.
Each line contains a single integer that denotes a value of **n ($1 \le n \le 10^6$)**. Input is terminated by a line containing a single zero.

Output

For each line of input produce one line of output. This line contains an integer that denotes the modulo **10000007 (or $10^7 + 7$)** value of **sndd(n)**.

Sample Input	Output for Sample Input
4 5 0	30 90

Problem D

Input: Standard Input
Output: Standard Output

Faketorial Hashing

Are you familiar with polynomial hashing? If you are not, all the better! You don't need to know what polynomial hashing is, the world is better off without it. I hate polynomial hashing so much that I found a new way to hash strings. It is called the ***Faketorial Hashing***.

First, let's define a function, *ord(ch)* = the position of *ch* in the alphabet + 1, where **ch** can be any lowercase letter. So, *ord(a) = 2, ord(b) = 3, ord(c) = 4, … ord(z) = 27*.

Let *fact(x)* be **x!** or the factorial of **x**. A few examples, *fact(1) = 1, fact(2) = 2, fact(3) = 6, fact(4) = 24, fact(5) = 120*, etc.

Given a string **S** of length **N,** consisting of lowercase letters only, the ***Faketorial Hashing*** of **S**, is defined as below:

fake_hash(S) = fact(ord(S[0])) × fact(ord(S[1])) × fact(ord(S[2])) × …… × fact(ord(S[N - 1]))

In other words, it is the product of the factorial of the **ord()** value of all the characters in **S**. (That's right, no modulus! Unlike the lame polynomial hashing.)

Not only that we have a new hashing mechanism in place, but we would also like to crack this now. Given a string S_1 consisting of lowercase letters only, your task is to find **a different** string S_2 consisting of lowercase letters**,** such that, **fake_hash(S_1) = fake_hash(S_2)** and $S_1 \neq S_2$.

If there are multiple possible choices for S_2**,** you need to find the **lexicographically smallest one,** or output the word **"Impossible"** without quotes, if it is not possible to find such a string.

Input

The first line contains an integer **T (1 ≤ T ≤ 3000),** denoting the number of test cases. Each test case contains the string S_1 **(1 ≤ $|S_1|$ ≤ 30)** consisting of lowercase letters only.

Output

For each test case, output the case number followed by the required output. Please refer to the sample input/output section for the precise format.

Constraints:

Except for the sample, the following constraints will hold:

1 ≤ $|S_1|$ ≤ 5, for **90%** of the test cases
1 ≤ $|S_1|$ ≤ 15, for **99%** of the test cases

Sample Input

```
10
tourist
petr
mnbvmar
bmerry
xellos
sevenkplus
dragoon
zzz
snapdragon
zosovoghisktwnopqrstuvwxyzoos
```

Output for Sample Input

```
Case 1: aaaaabbdnstttu
Case 2: aqst
Case 3: abmmnrv
Case 4: aaabbnrry
Case 5: aaaaaaadddlnuz
Case 6: aaaaaaabbddddnquuuz
Case 7: aaaaaaaaaaaaabdnnnt
Case 8: Impossible
Case 9: aaaaaaaaaabdnnnpst
Case 10: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaffffjnnnnqqttttuuuuxxzzzzzz
```

Problem E

Input: Standard Input
Output: Standard Output

Helping the HR

The University of **STU** has imposed some strict attendance rules for the faculty members. You are now being asked to write a program that will help the HR Department to monitor the monthly attendance records of its faculty members. The conditions are as follows:

1. A faculty member can be involved in either Day Shift or Evening Shift in a single day.
2. Faculty members involved in day shift classes in a day must come to the office by 9:30 AM and stay for at least 8 hours. However, hours spent in the office before 8:30 AM won't get counted.
3. Faculty members involved in the evening shift classes in a day must come to the office by 12:30 PM and must stay in the office for at least 9 hours. However, hours spent in the office before 8:30 AM won't get counted.
4. If a faculty member is late or fails to maintain required hours (8 or 9 hours based on involvement in day or evening shift classes) for more than three times in a month, he/she will be issued a show cause letter. Otherwise, 1 point will be deducted per late attendance or for failing to maintain required hours but no show cause letter will be issued.
5. If a faculty member is late and fails to maintain required hours in a single day, it will be counted as one point deduction. In other words maximum one point will be deducted for violation of more than one rule in a single day.

Given the number of days to be considered for attendance in a month, and the entry and exit time for each of those days for a faculty member, you will have to report one of the following about that employee:

1. "All OK"
2. "1 Point(s) Deducted"
3. "2 Point(s) Deducted"
4. "3 Point(s) Deducted"
5. "Issue Show Cause Letter"

Input

The input file contains several test cases.

The first line of each test case contains an integer **N (0<N<31)**, which denotes the total number of days in a month that needs to be considered. Each of the next **N** lines contains the description for a day. The description follows the following format:

`S:h1:m1:s1:h2:m2:s2`

Input is terminated by a case where **N=0**.

The value of **S** can be either 'D' or 'E' (Without the quotes), 'D' means that on this day the faculty member is involved in only day shift classes and 'E' means that the faculty member is involved in only evening shift classes. h1:m1:s1 denotes the entry time of the faculty member and h2:m2:s2 denotes the exit time in 24-hour format. You can assume that the times will be valid ($0 \le h_1, h_2 < 24$, $0 \le m_1, m_2, s_1, s_2 < 60$) and entry time will never be after exit time.

Output

For each set of input produce one line of output. This line contains any one of the lines "All OK", "1 Point(s) Deducted", "2 Point(s) Deducted", "3 Point(s) Deducted", "Issue Show Cause Letter" (Without the quotes)

Sample Input	Output for Sample Input
3 D:8:30:00:17:30:20 D:9:30:01:17:30:20 E:11:30:01:20:31:00 3 D:8:30:00:16:30:00 D:9:30:00:17:30:20 E:11:30:01:20:31:00 3 D:8:30:00:17:30:20 D:9:30:01:17:30:20 E:11:30:01:20:00:00 4 D:8:00:00:16:15:20 D:9:30:01:17:30:00 E:11:30:01:20:00:00 E:11:30:01:20:00:00 0	1 Point(s) Deducted All OK 2 Point(s) Deducted Issue Show Cause Letter

Problem F

Input: Standard Input
Output: Standard Output

Path Intersection

Do you know trees? They have branches and leaves. Their roots stay inside the ground to make a solid base and draw water from the soil. People get oxygen and foods from them.

In graph theory **Tree** is known as a connected graph consisting of **N** nodes and **N-1** edges. The nodes are denoted by integers from **1** to **N**. So there is exactly one path between any two nodes. The root stays on top making the whole tree hang under it. People ask silly and nerdy questions about it. Speaking of one such silly questions, do you happen to know how to find out the number of common nodes among all the **K** given paths in a tree?

Input

There will be **T (1 ≤ T ≤ 50)** test cases. Each case will start with an integer **N (1 ≤ N ≤ 10000),** representing the number of nodes in the tree. Then, **N-1** lines will follow, each having two integers **u** and **v (1 ≤ u, v ≤ N),** representing an undirected edge from **u** to **v**. The next line will contain an integer **Q (1 ≤ Q ≤ 10000),** denoting the number of queries. Each query will start with an integer **K (2 ≤ K ≤ 50)** denoting the number of paths. Then, **K** lines will follow each having two integers **a** and **b (1 ≤ a, b ≤ N),** representing a path from **a** to **b**. For the entire input file, **∑N ≤ 500000, ∑Q ≤ 500000, ∑K ≤ 500000**. The dataset is huge, so use faster I/O methods.

Output

For each test case, print the case number in a line. Then for each query, print an integer in a line denoting the number of common nodes among all the **K** given paths.

Sample Input

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

Output for Sample Input

```
Case 1:
2
1
Case 2:
0
```

Problem G
Techland

Input: Standard Input
Output: Standard Output

Rahim is very much enthusiastic about technologies. He is always ready to buy new gadgets and stuff. But before buying anything, he always watches a lot of reviews and gathers information about that specific stuff. The country **"TechLand"** where Rahim lives, doesn't believe in a competitive market. That's why all the products in TechLand are of the same price. When Rahim wants to buy anything, he visits the closest shop from his position and buys the product.

TechLand has **N** cities which are connected with **N-1** bi-directional roads. The cities are denoted by integers from **1** to **N**. In this country, sometimes new companies come to do some business and sometimes they leave the country. So, the shop from where Rahim will buy his gadgets is not always fixed.

In this situation, you will be given **Q** events. The events can be one of the following:

1. **X L R:** The company **X** is going to start new business and they will open shops at the cities labeled with **L, L+1, L+2, …, R-2, R-1, R (L ≤ R).** If the company already exists in the country, then they will shut down all the shops opened previously and will open new shops in these cities (whose labels are between **L and R inclusive**). In a single city, there can be multiple shops, but no company will have multiple shops in a city.
2. **X**: Company **X** is leaving TechLand. It is guaranteed that the company **X** was doing their business before leaving the country. In the future, the company may or may not come back again.
3. **C M P_1 P_2 … P_M**: Rahim wants to buy a new gadget. Currently, he is at city **C**. He has also prepared a preferred list of companies for purchasing the gadget. The list contains **M** companies **P_1, P_2, P_3, … , P_M**. To buy his gadget, Rahim will travel to the closest city from **C** where at least one of the preferred companies has a shop. It is guaranteed that each of the companies in Rahim's preferred list has come at least once in the country to establish their business. But it is not guaranteed that they are still continuing their business. Sometimes it may happen that Rahim is unable to find any shop of those **M** companies. In that situation, you will also need to report that.

Rahim is not that good at finding the closest city he needs to travel. You guys are good at programming! So help him to figure out the shortest distance he will need to travel to buy the gadget for each of the **type 3** events.

Input

The first line of the input will contain a single integer **T (1 ≤ T ≤ 10)** denoting the number of test cases. Each test case contains several lines. The first line of each test case will contain an integer **N (1 ≤ N ≤ 50000)** which denotes the number of the cities in TechLand. Each of the next **N-1** lines contains two integers **U V (1 ≤ U, V ≤ N and U≠V)** which means there is a bidirectional road connecting cities **U** and **V**.

After that the next line will have a single integer **Q (1 ≤ Q ≤ 100000)** representing the number of events. Each of the next **Q** lines describes a single event. Each event line starts with an integer **E (1 ≤ E ≤ 3)** denoting the event type.

If **E = 1**, then it will contain three more integers **X, L,** and **R (1 ≤ X ≤ 100000** and **1 ≤ L ≤ R ≤ N)**.
If **E = 2,** then the line will contain another integer **X (1 ≤ X ≤ 100000)**.

If **E = 3**, the line will start with two integers **C** and **M (1 ≤ C ≤ N** and **1 ≤ M ≤ 100000)**. After these two integers there will be **M** more integers $P_1, P_2, P_3, \ldots, P_M$ **(1 ≤ P_i ≤ 100000)**.

Each event is described elaborately in the description above. In a single test case the sum of **M** over all the events will not exceed **100000**.

Note: Input file is huge. Please use faster I/O.

Output

The first line of each test case should contain the case number in the format: **"Case Y:**" without quotes, where **Y** denotes the test case number. After that for each event of **type 3**, a single line should contain the minimum distance Rahim will need to travel in order to buy the gadget. If it is not possible to find such a city where Rahim should travel then just output **-1** instead.

Sample Input	Output for Sample Input
2 6 1 4 1 5 1 6 5 2 5 3 4 1 1 2 3 3 5 1 1 2 1 3 4 1 1 2 1 2 2 1 100000 1 1 3 2 1 100000	Case 1: 1 -1 Case 2: 1

Problem H

Input: Standard Input
Output: Standard Output

Tile Game

We have a board and there are some square shaped tiles inside it. The board and the tiles look similar to the picture on the right:

As you can see the board has some borders, and the tiles inside the board are solid square shaped. If you tilt the board in some direction, all the tiles will try moving to that direction until they hit the board boundary or some other tiles. We can tilt the board in four directions: up, down, left and right. For example, let's consider following 3×3 board with 4 tiles inside. The tiles are marked from 1 to 4:

Initial state

		3
1	4	
	2	

For your convenience, we have shown below the resulting board state after tilting the initial board in each of the four directions.

upward	1 4 3 2	leftward	3 1 4 2
downward	4 1 2 3	rightward	3 1 4 2

We call a board state **stable** if the tiles don't move if we tilt it leftward or downward. That is, the tiles will be cluttering around the bottom-left corner. You are given an (R × C) shaped **stable** board (R rows and C columns). There are some tiles inside it and each of these tiles is marked with **distinct colors**. You can tilt the board in any direction any number of times. You need to find the number of distinct stable board states that you can arrive at.

Two stable boards are distinct if in one board we have a color at one cell but in the other board, we have the same color in some different cell.

Input

In the first line of the input, you are given the number of test cases, **T (1 ≤ T ≤ 100)**. Hence follows T test cases.

Each test case starts with two positive integers **R** and **C (1 ≤ R, C ≤ 200)**. Then following R lines contain C characters each, denoting an R×C shaped **stable** board. Each character is either a dot (.) or a hash (#). A hash denotes a tile. It's guaranteed that there is at least one tile in the board.

Output

For each test case, print the case number followed by the answer of the test case. Since the answer can be large, print modulo **78294349** of it. For the specific output format, please consult the sample input/output.

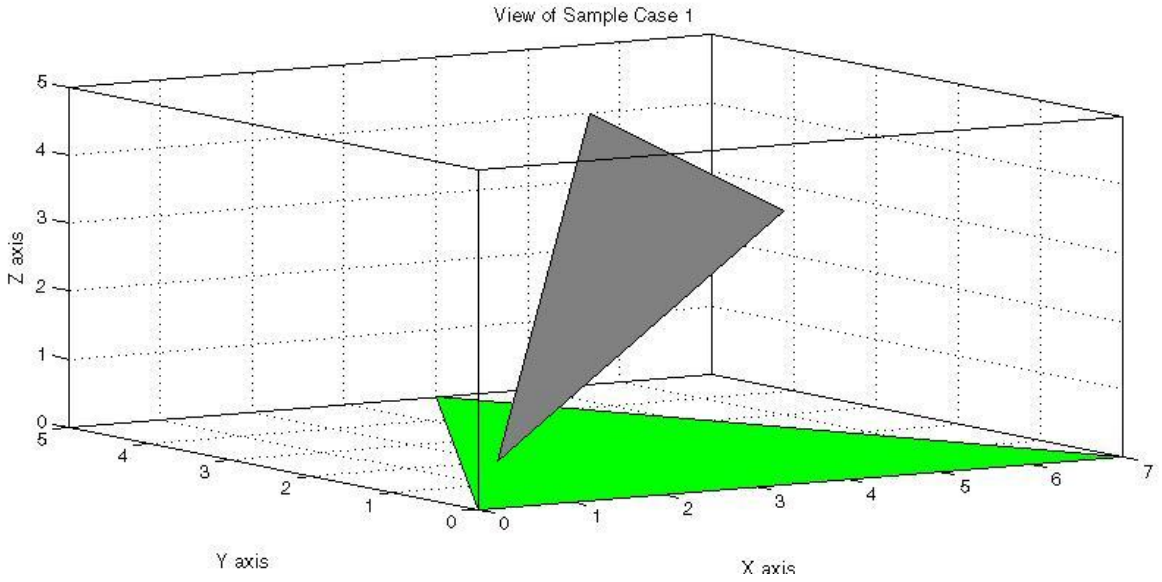
Sample Input	Output for Sample Input
3 2 2 .. #. 2 2 #. #. 18 13 ..... ##..... ##..... ####..... ####..... #####..... #####..... #######..... #######..... #########.... #########.... ##########... ###########.. ###########.. ###########.. ############. ############# #############	Case 1: 1 Case 2: 1 Case 3: 27457584

Problem I

Input: Standard Input
Output: Standard Output

Triangles

In this problem, you are given the vertices of two triangles in **3D** space. Determine the minimum distance between the plane segments bounded by the edges of the triangles. See the image below and samples for a clear understanding.



Input

The first line will contain a positive integer **T (1 ≤ T ≤ 10000)**, the number of test cases.

For each test case, two lines will contain **9** space separated integers each, denoting **3** coordinates of the vertices of the two triangles. The triangles will have a **non-zero** area. The absolute value of the coordinates will not exceed 10^5.

Consecutive test cases will be separated by a new line.

Output

Print the distance. Absolute error less than 10^{-6} will be ignored. Please check the sample for details.

Sample Input	Output for Sample Input
2 0 0 0 7 0 0 4 5 0 2 2 0 3 2 5 6 3 3 0 0 0 7 0 0 4 5 0 2 2 1 5 4 10 5 4 11	Case 1: 0.000000000 Case 2: 1.000000000

Problem J

Input: Standard Input
Output: Standard Output

VAT Man

The people of Wakanda pay income tax every year to fund all the initiatives of the Government. To facilitate all the newly introduced radical development plans however, the Government needs a lot more fund. On top of the regular income tax, the citizens of Wakanda are now required to pay an additional 15% on any purchase. This extra amount is going to be collected as VAT (Vibranium Acquisition Tax). Wakanda needs your help to calculate the price of any product; Wakanda wants you to be their VAT Man!

Input

The input consists of several test cases. The first line gives you the number of test cases, **T (T < 100)**. For each of the next **T** lines, you'll get exactly one integer -- the price of the product, **P (1 ≤ P ≤ 10000)**.

Output

For each test case, you only need to print the total price of the product including 15% VAT. Print your answer rounded to two decimal places.

The following table shows you code snippets in C, C++, and Java to calculate the total price (including 15% VAT) of a product that's worth 100 WD (Wakandian Dollar). It also shows you how to print your answer rounded to two decimal places.

C	C++	Java

#include <stdio.h>
…
 	
double price = 100;
double output = price * 1.15;
printf("%.2lf\n", output);
	#include <iostream>
	#include <iomanip>
	using namespace std;
	…
	
	double price = 100;
	double output = price * 1.15;
	cout << fixed << setprecision(2)
	  << output << endl;
		double price = 100;
		double output = price * 1.15;
		System.out.printf("%.2lf\n",
		  output);

Sample Input

```
2
100
1021
```

Output for Sample Input

```
115.00
1174.15
```