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

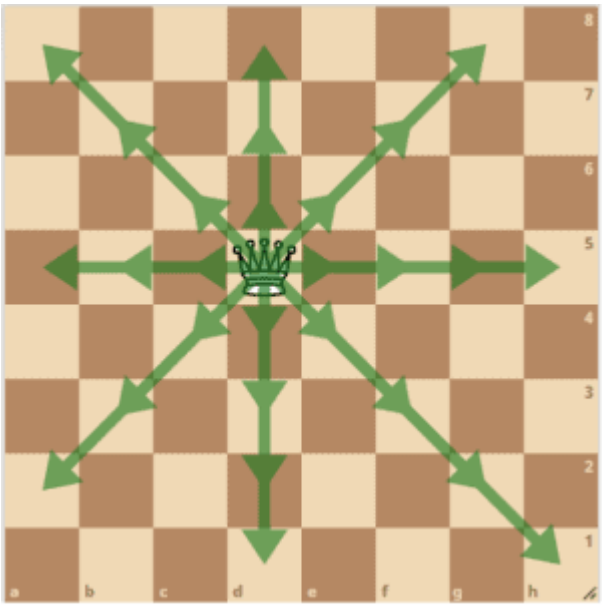
Print the case number followed by the answer to that test case in this format "**Case X: Y**" (without quotes), where **X** is the case number and **Y** is the number of ways for forming a pairdrome modulo **$10^9 + 7$.** Please see the sample for details.

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

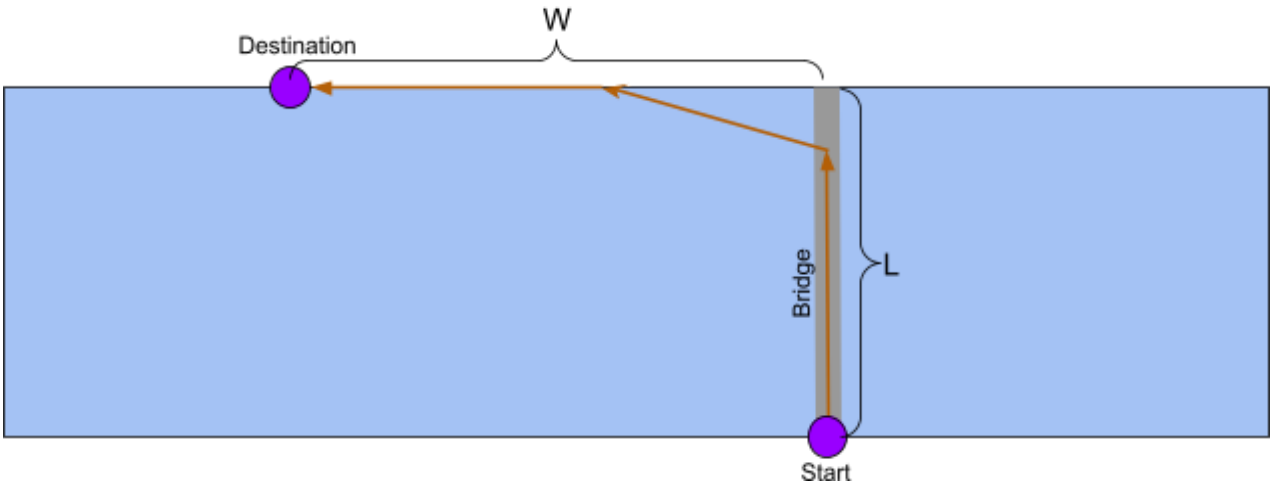
Please use fast input output for this problem.

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