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).**

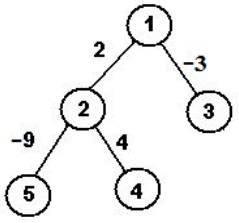
Sample Input	Output for Sample Input
2 5 2 3 4 1 4 4 1 3 1 1 3 2	Case 1: 24 Case 2: 0

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.

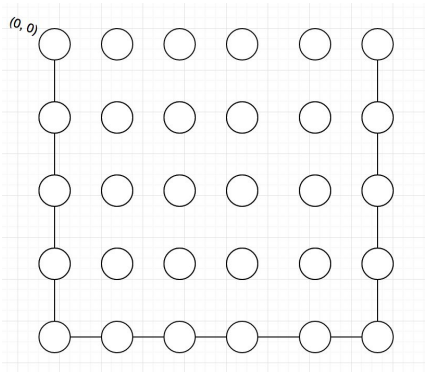
Sample Input	Output for Sample Input
1 5 5 5 3 8 2 0 1 12 1 9 7 5 1 3 2 4	Case 1: 48 16 41 31 47

Problem H

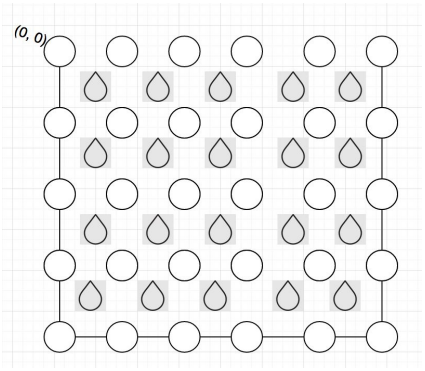
Droplets

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.

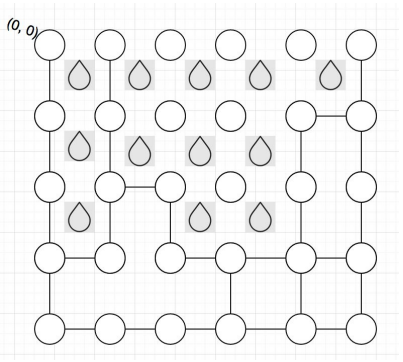
We have a matrix which consists of **N** rows and **M** columns.. Consider every cell of the matrix as a dot. The top left cell of the matrix is (0, 0). Initially it looks something like Picture 1.



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