ICPC Preliminary Dhaka Site 2024

Problem Set

A. Adhoc King

CPU: 1.00s Memory: 2048MB

Pebae has an integer n. For an integer sequence $a_1, a_2, \ldots, a_n$ an and an integer k, she defines $f(k)$ as $\left(a_1 \times a_2 \times \ldots \times a_k\right) \& \left(a_1 + a_2 + \ldots + a_k\right)$ — that is, $f(k)$ is the bitwise AND of the product and sum of the first k elements of the sequence.

An integer sequence is good if $f(i) \neq f(j)$ is satisfied over all pairs $1 \leq i < j \leq n$.

Now. Pebae wants to find a good sequence of length n such that all elements of the sequence are from 2 to $3n$ and the difference between the maximum and minimum element of the sequence is at most 10.

Help Pebae find such a sequence. It can be shown that such a sequence always exists under the given constraints.

Input Description

The first line contains a single integer $t\left(1 \leq t \leq 10^4\right)$ — the number of test cases.

The first and only line of each test case contains an integer $n\left(2 \leq n \leq 10^6\right)$.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output Description

For each test cases, print n integers — the integer sequence that satisfies the conditions mentioned in the statement. If there are multiple such sequences, output any.

Samples

Input	Output

3 3 2 6	5 7 5 3 2 11 12 3 2 9 12

Note

In the first test case,

- $f(1) = 5 \,\&\, 5 = 5$.
- $f(2) = (5 \times 7) \,\&\, (5 + 7) = 35 \,\&\, 12 = 0$.
- $f(3) = (5 \times 7 \times 5) \,\&\, (5 + 7 + 5) = 175 \,\&\, 17 = 1$.

Here, all the values of $f(k)$ are distinct, so the sequence is good.

B. Network Disentanglement

CPU: 3.00s Memory: 2048MB

Though Maryland is a remote area, luckily there are broadband internet connection services. Maryland is presented as a $2D$ geometric plane, where there exist N routers and M wire connections (straight line segments) that link pairs of routers located on the 2D plane. There are only two internet service providers, X and Y. As these connections are placed on a $2D$ geometric plane, it can happen that any two wires cross or touch each other. As a result, they face problems while providing service when wires of different ISPs cross or touch each other (they can handle when their own wires cross or touch each other). Now, they need your help to solve their problem. Your task is to find the minimum number of wires to remove such that no two wires of different ISPs can cross or touch each other, and you also have to find which wires to remove.

NB: Two wires cross or touch each other when the intersecting point(s) lies on both wires. So it can happen that the two wires lie on the same line. There can be multiple wires between any two routers. There won't be a wire with both endpoints connected to the same router.

Input Description

The first line will contain $T(1 \leq T \leq 50)$, the number of test cases. Each case will start with two integers $N(2 \leq N \leq 1000)$, $M(1 \leq M \leq 1000)$, where N denotes the number of routers and M represents the number of wire connections in Maryland. Each of the next N lines will contain two integers: $X_i\left(-10^9 \leq X_i \leq 10^9\right)$ and $Y_i\left(-10^9 \leq Y_i \leq 10^9\right)$, denoting the location of the i^{th} router. No two routers are situated in the same place. Each of the next M lines will contain three integers: $U_i\left(1 \leq U_i \leq N\right)$, $V_i\left(1 \leq V_i \leq N\right)$, and $I(1 \leq I \leq 2)$, where $U_i \neq V_i$, denoting the i^{th} wire link between routers U_i and V_i in the $2D$ plane, and I denotes whose wire it is ($I = 1$ for X, $I = 2$ for Y).

Output Description

For each case, you need to print **"Case #T:"** in the first line (without quotes), where T is the test case number. The second line contains an integer denoting the minimum number of wires to remove such that no two wires of different ISPs can cross or touch each other. The third line of each case contains space-separated wire link numbers (I-based) to remove. **The third line will be an empty line if you don't have to remove a wire link.**

Samples

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

C. Watering the Flowerbed

CPU: 1.00s Memory: 2048MB

In the fertile and magical lands of Numeria, a legendary garden is entrusted to your care. This garden contains N **flowerbeds**, each filled with delicate and precious flowers that require precise watering to stay healthy. However, these flowers are sensitive, and maintaining their well-being is no simple task.

Each flowerbed has two critical watering requirements:

For, i^{th} flowerbed, if it is not watered within X_i days since its last watering, the plants will dry out and die.

Also, if the i^{th} flowerbed is watered again within Y_i days since its last watering, the excess water will harm the roots and plants may die.

All the flowers were watered today. You have to take care of the garden for K more days. Find out the minimum total number of **times** you need to water to ensure all flowerbeds remain healthy after K **days**. On any day, if you water multiple flowerbeds, you have to count them separately.

Input Description

The input consists of multiple test cases.

The first line contains an integer $T(1 \leq T \leq 25)$, the number of test cases. For each test case, The first line contains two integers $N(1 \leq N \leq 1000)$ and $K\left(1 \leq K \leq 10^9\right)$ where N denotes the number of flowerbeds and K denotes the number of days you need to take care of the flowerbeds. The next N lines each contain two integers, X_i, Y_i where $\left(1 \leq Y_i < X_i \leq 10^9\right)$.

Output Description

For each test case, print the case number followed by the total number of times you need to water.

Samples

Input	Output
1 2 10 3 1 6 3	Case 1: 4

D. Red Apple, Green Apple

CPU: 3.00s Memory: 2048MB

You have N boxes, each box has one red apple and one green apple. Each of the apples have a specific weight. You want to pick exactly K apples, so you select K boxes randomly. From each of the selected boxes, you either pick the red apple or the green apple with the same probability. After selecting the apples you sum up their weights.

For each K from 1 to N, find the expected total apple weights when picking exactly K apples. We can express the expected value as $\frac{P}{Q}$ $(Q \neq 0)$, answer the value of $\left(P \cdot Q^{-1}\right)$ $modulo$ 998244353.

Input Description

The first line contains an integer $N(1 \leq N \leq 300000)$. The second line contains N integers denoting the weights of the red apples of each box. The third line contains N integers denoting the weights of the green apples of each box. The weights are positive integers and at most 1,000,000,000.

Output Description

Print N space separated integers in a single line, where the i^{th} integer represents the answer for $K = i$.

Samples

Input	Output
10 3 1 5 3 1 3 5 4 2 2 3 3 1 2 2 5 3 5 1 5	848507703 698771053 549034403 399297753 24956110...

E. Master Evenius and Oddius

CPU: 1.00s Memory: 2048MB

In the mystical land of Numeria, two legendary masters, **Evenius** and **Oddius**, are locked in an eternal duel to determine who will claim the throne of the Stone Kingdom. Their battleground is a pile of stones. This pile has N **magical stones**. Each stone possesses an ancient power, and only by playing optimally a master can ensure victory.

The rules of the duel are as follows:

Evenius can only remove an **even** number (2, 4, ... etc) of stones on his turn, but not more than $\left\lfloor \frac{N}{2} \right\rfloor$ stones. Oddius can only remove an **odd** number (1, 3, ... etc) of stones on his turn, but not more than $\left\lfloor \frac{N}{2} \right\rfloor$ stones. Note that, here N represents the initial number of stones and $[x]$ denotes the floor function.

These two masters take turns removing stones from the pile, starting with a designated player. The master who is unable to make a valid move (can't remove any stone) loses the duel, and the other master is crowned the ruler of Numeria.

Both masters are strategists and will always play optimally to ensure their victory. Your task is to predict the outcome of this legendary battle and determine who will emerge victorious.

Input Description

The input consists of multiple lines. The first line contains an integer $T(1 \leq T \leq 10,000)$ representing the number of test cases. Each of the next T lines contains two integers: $N\left(1 \leq N \leq 10^{16}\right)$ denoting the initial number of stones in the pile and P representing the starting player, $P = 1$ represents Oddius and $P = 2$ represents Evenius.

Output Description

For each test case, print the test case number in the format **"Case X: "**, followed by the name of the master who will win the crown if both of the players play optimally. See the sample Input/Output section for more clarity.

Samples

Input	Output
3 1 1 2 2 10 2	Case 1: Evenius Case 2: Oddius Case 3: Oddius

F. Three Quick Brown Foxes Jump Over a Lazy Chicken

CPU: 1.00s Memory: 2048MB

Three clever foxes each guard a side of a triangular field. This field is fenced on all sides, and within its boundary roams a cautious chicken. Every day, the chicken roams freely around the field, covering every point of the space within the triangle, yet never touching the fenced edges that form its boundary.

Eager to catch the chicken, the foxes devise a plan. First, each fox must find the perfect spot along their respective side of the triangle to take position. Then, they must identify a point inside the triangle to place a trap. Their goal is to create a setup where they can jump from their positions to this trap, reaching it at the same time to catch the chicken, while each fox covers the shortest possible distance at the same speed. Note that no two foxes will share the same side of the triangle. The foxes can't take positions on the vertices of the triangle.

The challenge is now clear: with only the lengths of the triangle's sides to guide them, the foxes must figure out the precise positions and shortest synchronized jump that will lead them to the trap and, finally, to the elusive chicken.

You will help the foxes figure out the shortest possible distance of the jump needed, given the lengths of the sides of the triangular field.

Input Description

The first line will contain a single integer $T(1 \leq T \leq 1000)$. Each test case will have three integers X, Y, and $Z(1 \leq X, Y, Z \leq 1000)$, denoting the lengths of three sides of the triangular field. It is guaranteed that the side lengths will make a valid triangle.

Output Description

For each test case, let the length of the shortest possible jump is L. You have to print the square of L as an irreducible fraction. An irreducible fraction is a fraction in which the numerator and denominator are integers that have no common divisors other than 1.

Samples

Input	Output
2 57 13 110 103 17 110	32/1 1176/23

G. Travel on the grid

CPU: 10.00s Memory: 2048MB

You are given a grid of a size $N \times M$. You need to travel from cell (1, 1) to (N, M). You can travel to any of the 8 adjacent cells from your position. There are mines in some cells. When any mine explodes, it can kill anyone standing on its cell or anyone standing on the 4 other adjatimescells of the mine (top, bottom, left, right). Stepping into any of these five cells will trigger the mine and it explodes.

You can build and use as many diffusers as you need to avoid the mines. Also, you can use a single diffuser as many times as you need. You need to place the diffuser directly on the cell where a mine is located to neutralize it. Once neutralized, the mine poses no further threat and it will never explode in the future. Building a diffuser and reusing a diffuser both have its own cost.

At any time, while in a safe cell (one without a mine and not within the blast radius of any adjacent mines), you can build a diffuser and place that in any of the 8 adjacent cells from your current position.

In order to reuse a diffuser, you will need to travel to the cell where it is located. Then you can take the diffuser and travel with it. When you are traveling with a diffuser, you will not be able to build a new one or pick up another diffuser. At any time, you can deploy a carried diffuser in any of the 8 adjacent cells of your current position if that cell contains a mine. You can't drop a diffuser in a cell if it doesn't have any mines. Each time you reuse the diffuser, there is a fixed cost associated with it. Additionally, each cell on the grid has its own specific cost when you move from that cell to an adjacent one while carrying the diffuser. Therefore, the total cost of transporting and deploying the diffuser will be the sum of this fixed cost and the individual cell costs incurred during travel.

Now you need to find the cheapest way to reach the cell (N, M). **It is guaranteed that no mine in the grid will attack cell (1, 1). Also no single cell will be attacked by two different mines.**

Input Description

The first line will contain a single integer $T(1 \leq T \leq 1000)$. First line of each test case will contain four integers $N(2 \leq N \leq 7)$, $M(2 \leq M \leq 7)$, $X\left(1 \leq X \leq 10^9\right)$ and $Y\left(1 \leq Y \leq 10^9\right)$. Here N and M represent the dimension of the grid $(N \times M)$ while X is the cost for budding a diffuser and Y is the fixed cost for reusing a diffuser . The following N lines will contain a string, each of them will be of length M. Each character of the strings will be either '.' (ASCII code 46) or

‘#’ (ASCII code 35). The character ‘.’ denotes a safe place and the character ‘#’ denotes the cell has a mine on it. Following these, there will be N additional lines. Each line will contain M integers $V_{ij}\left(0 \le V_{ij} \le 10^9\right)$ where each V_{ij} represents the associated cost of cell (i, j) for moving to an adjacent cell while carrying a diffuser.

Output Description

For each test case, you need to print the answer in the format **"Case A: B"** where A is the case number and B is the cheapest cost to reach (N, M) from (1, 1).

Samples

Input	Output
2 3 4 10 5 .... .#.. ...# 0 0 0 0 0 1 2 3 0 0 0 0 3 4 10 10 ..#. .... #..# 0 0 1 0 0 0 0 0 1 0 0 0	Case 1: 16 Case 2: 20

H. GonaGuni2

CPU: 6.00s Memory:2048MB

You are given a $N \times M$ grid and K colors. You need to color all the cells from those K colors such that no side adjacent cell has the same color. Find the number of ways to color it modulo 998244353.

Input Description

First line contains a single integer T representing the test cases. For each test case there will be a single line containing 3 space separated integers N, M and K.

Output Description

For each test case print a single line representing the answer modulo 998244353.

Constraints

- $1 \leq T \leq 7$
- $1 \leq M \leq 7, 1 \leq N \leq 10^7, 1 \leq K \leq 10^7$
- Sum of M over all test cases ≤ 7

Samples

Input	Output
1 10 4 8	422121017