Inter University Programming Contest (IUPC)

Hosted By
Department of Computer Science and Engineering
United International University

Supported by

You get 11 problems, 17 pages and 300 minutes.

Problem A. A Boring Number Game

Alice and Bob are bored on a rainy day with nothing fun to do. So, they decide to make up a game to pass the time. The rules are simple: Alice and Bob will take turns building a list which starts out **empty**, by adding numbers to it. They have exactly n turns to play (counting both their turns).

Alice always goes first. On her turn, she picks a random integer from $[1, k]$ uniformly and adds it to the beginning of the list. Bob, on the other hand, takes his turn by picking a uniform random integer from the same range and adding it to the end of the list. This back-and-forth continues until all n turns are used.

Once they're done building the list, Alice and Bob look at the list and remove any consecutive duplicates. For example, if their list looks like this after n turns: $\{1, 1, 2, 3, 3, 3, 4, 3, 3\}$, it becomes: $\{1, 2, 3, 4, 3\}$ after removing the consecutive duplicates.

If the length of the final list is odd, Alice wins. Otherwise, Bob wins.

But simply winning isn't enough for Alice and Bob. They're curious about the expected length of the final list after cleaning up the duplicates. Why not take a moment to figure that out as well? Since you're the one who can tackle all their math problems, they turn to you for the answer. Can you help them out?

Given n and k, find the expected length of the final list after cleaning up the consecutive duplicates. The answer can be expressed as $\frac{P}{Q}$, $Q \neq 0$. You need to output the value: $P \cdot Q^{-1} \mod 998244353$, where Q^{-1} is the modular multiplicative inverse of Q modulo 998244353.

Input

The first line contains an integer T ($1 \leq T \leq 10^5$) — the number of test cases. Each of the next T lines contains two integers n, k ($1 \leq n, k \leq 10^6$).

Output

Print a single integer — the value of $P \cdot Q^{-1} \mod 998244353$.

Sample Input	Sample Output
3 1 1 2 1 2 2	1 1 499122178

Explanation

For $n = 2$ and $k = 1$ there is only one possible list: $\{1, 1\}$. Hence, the expected value is 1.

For $n = 2$ and $k = 2$ all the possible lists are: $\{1, 1\}, \{1, 2\}, \{2, 1\}, \{2, 2\}$. We can see that the expected value is $\frac{3}{2}$. Thus, the final answer would be $3 \cdot 2^{-1} \mod 998244353 = 499122178$

Problem B. Bijgonit

Given two integers X and Y, find the value of $\gcd(X^6 - Y^6, X^4 - Y^4)$.

Input

The first line of the input contains a single positive integer t ($1 \leq t \leq 5 \times 10^5$) — the number of test cases. Then t test cases follow.

Each test case contains two integers in a line — X and Y ($0 \leq Y < X \leq 5000$).

Input file is large. Please be sure to use faster input/output methods.

Output

For each test case, output one integer in a line — $\gcd(X^6 - Y^6, X^4 - Y^4)$.

Sample Input	Sample Output
2 5 3 5000 4999	16 9999

Explanation

$\gcd(A, B)$ is the greatest common divisor of A and B. $g = gcd(A, B)$ satisfies the following properties:

- $g \mid A$ and $g \mid B$.
- If $h \mid A$ and $h \mid B$, then $g \geq h$.

Here, $a \mid b$ means that there exists an integer c such that $b = ac$.

In the first test case, $\gcd(5^6 - 3^6, 5^4 - 3^4) = \gcd(15625 - 729, 625 - 81) = \gcd(14896, 544) = 16$

Problem C. Magnetic Bond Optimization

In a distant land, magnetic particles with varying strengths and polarities are aligned in a straight line. Each particle has a value (strength) and a polarity: positive or negative. Two particles with opposite polarities can form a bond between them, while particles with the same polarity repel each other and do not form any bond. There is an integer array $A_1, A_2, ..., A_i, ..., A_N$ of N integers, which represents a particle chain. The sign of an element A_i denotes its polarities and the absolute value of A_i denotes its strength. The bond strength between two adjacent particles A_i and A_j having opposite polarities is calculated as: $min(|A_i|, |A_j|)$. If two adjacent particles have the same polarity, they do not contribute to the bond strength. The bond strength of a particle chain is the summation of bond strengths of all pairs of adjacent particles.

You are given Q queries. For each query, you are given an integer K, where you will select a sub-chain of length K from the given N particles such that the bond strength of the sub-chain is maximized. A sub-chain is a contiguous subarray of the given array A. For each query, you need to find the maximum total bond strength of any sub-chain of length K.

Input

The first line will contain T ($1 \leq T \leq 10$), the number of test cases.

The first line of each test case contains two integers N ($2 \leq N \leq 10^5$) and Q ($1 \leq Q \leq 10$): the number of magnetic particles and the number of queries.

The second line of each test case contains N integers $A_1, A_2, ..., A_N$ ($-10^9 \leq A_i \leq 10^9, A_i \neq 0$): the values of the particles. Positive values represent positive particles, and negative values represent negative particles.

The next Q lines of each test case contain a single integer K ($2 \leq K \leq N$): the length of the sub-chain.

Output

For each test case, output a single integer for each query on a separate line which denotes the maximum bond strength of any sub-chain of length K

Sample Input	Sample Output
1 6 1 5 -7 8 -10 6 4 4	21

Explanation

One possible sub-chain of length K = 4: [5, -7, 8, -10]
Pairwise bond strengths: $\min(|5|, |-7|) = 5$, $\min(|-7|, |8|) = 7$, and $\min(|8|, |-10|) = 8$
Total strength for this sub-chain: $5 + 7 + 8 = 20$.

Another possible sub-chain of length K = 4: [-7, 8, -10, 6]
Pairwise bond strengths: $\min(|-7|, |8|) = 7$, $\min(|8|, |-10|) = 8$, and $\min(|-10|, |6|) = 6$
Total strength for this sub-chain: $7 + 8 + 6 = 21$.

The last possible sub-chain of length K = 4: [8, -10, 6, 4]
Pairwise bond strengths: $\min(|8|, |-10|) = 8$, $\min(|-10|, |6|) = 6$, and $\min(|6|, |4|) = 0$
Total strength for this sub-chain: $8 + 6 = 14$.

Optimal sub-chain: [-7, 8, -10, 6], with maximum total bond strength = 21.

Problem D. Daripalla

Mr Ladim, an honest Bangladeshi shopowner, uses a classic "daripalla" to measure weight of items. In case you don't know, you use a daripalla by putting the item to weigh on one side and measuring weights, which are called "batkhara", on one or both sides. For example, if you see that item on left side and 5 kg batkhara on right side balance the daripalla, you deduce the item is 5 kg. And if you see that item and 2 kg batkhara on left side and 5 kg batkhara on right side balance the daripalla, then the item is 3 kg.

Now Mr Ladim had n distinct batkharas in his possession (not necessarily all different in weight). Let's say these batkharas are $b_1, b_2, \cdots, b_n$ where ith batkhara has weight b_i. Using those batkharas, Mr Ladim could measure a range of possible weights. In fact, to be sure of exactly which weights he could measure, he sat down and recorded all those weights. Specifically, he tried out all possible combinations of the batkharas (**always keeping the target item with positive weight on left side**) and for each combination, wrote the target weight he could measure. Formally speaking, two combinations are different if the set of batkhara indices which are put on two sides do not match. **Note that same target weight may be measurable in different combinations and he recorded it each time.**

Sadly, Mr Ladim can't find the batkharas. So he needs to have them remade. He wants to make an exact replica of his previous batkharas, but he only has the list of measurable weights he had made earlier. Can you reconstruct original batkhara weights from the list?

Input

First line of input will contain an integer T denoting the number of independent test cases. Each test case will start with an integer k denoting the number of weights in Mr Ladim's list. Next line will contain k integers $a_1, a_2, \cdots, a_k$ separated by space. These are the weights he recorded.

- $1 \leq T \leq 5 \times 10^6$
- $1 \leq k \leq 5 \times 10^6$
- Sum of k over all test cases $\leq 5 \times 10^6$
- $1 \leq a_i \leq 10^9$

Output

For each of the test cases, print an integer n in first line. If it's not possible to reconstruct the original batkharas, n should be -1. Otherwise n should be the number of batkharas. And in next line put the batkhara weights separated by space in **non-decreasing order**. **Do not print any trailing space in any of the output lines.**

Sample Input	Sample Output
3 4 1 3 4 2 12 1 1 3 2 2 4 1 1 3 1 3 5 4 1 4 6 6	2 1 3 3 1 2 2 -1

Note

For second case, the batkhara weights are $a = 1, b = 2, c = 2$ then the weights in input are measured in following ways:

- a on right side measures 1
- a, b on right side measures 3
- b on right side measures 2
- c on right side measures 2
- a, c on right side measures 3
- b, c on right side measures 4
- a, b, c on right side measures 5
- b on right side and a on left side measures 1
- c on right side and a on left side measures 1
- b, c on right side and a on left side measures 3
- a, b on right side and c on left side measures 1
- a, c on right side and b on left side measures 1

Problem E. Escape Plan II

After successfully escaping from the first city with your cybertruck, you found yourself in a bigger and more dangerous city. This city is attacked by level 2 zombies. The city has N important junctions numbered from 1 to N. i^{th} junction has a fuel station with a limited amount of fuel F_i. N junctions are connected with $N-1$ roads (you can reach any junction from another junction if you have enough fuel). i^{th} road connects junction u_i and v_i bidirectionally and has a fuel consumption value of w_i, means your cybertruck needs w_i amount of fuel to travel the road.

Currently, you are at the fuel station of junction S. You have already refueled at the fuel station at junction S. You can escape through any outer junction that is connected with less than or equal to one junction. Formally, the city structure forms a tree and you can escape through a leaf node. You can not escape through S even if it's a leaf, because that way leads to the previous city.

The zombies are so dangerous and overwhelming that you can't afford to visit the same road twice. And stopping to refuel is even more dangerous. So, you need to find a route that requires the fewest stops.

There is some fuel left in your cybertruck from the previous city. This fuel information is provided to you through queries. You will be given Q queries, where each query provides an initial fuel amount from the previous city, A_i. For each query, your task is to calculate the fewest number of additional stops (excluding S) required to escape the city.

Note that you always start at the junction S and always take fuel from this junction. So, while leaving junction S for i^{th} query, you will have $A_i + F_s$ amount of fuel in your cybertruck.

Although your cybertruck has infinite fuel capacity, if it runs out of fuel in the middle of a road, you will surely be killed. So, solve the problem seriously.

Input

The first line of the input will contain an integer $T(1 \leq T \leq 10)$ that indicates the number of test cases. Then T test cases follow.

The first line of each test case will contain three integers $N(2 \leq N \leq 2000)$, $S(1 \leq S \leq N)$, and $Q(1 \leq Q \leq 10^5)$ - the number of junctions, the starting junction, and the number of queries.

The second line of each test case will contain N space-separated integers $F_1, F_2, \ldots, F_N(1 \leq F_i \leq 10^9)$ - amount of fuel in each junction.

The next $N-1$ line of each test case will contain three space-separated integers u_i, v_i and $w_i(1 \leq u_i, v_i \leq N, 1 \leq w_i \leq 10^9)$ describing each road.

The next Q lines contain one integer $A_i(0 \leq A_i \leq 10^9)$ - the amount of initial fuel.

Input file is huge. Use faster I/O.

Output

For each of the test cases, print the Q lines, the fewest number of additional stops (excluding the starting junction) needed to escape the city when you have A_i initial fuel. If it is impossible to escape the city, print `"impossible"`(without quotes).

Sample Input	Sample Output
1 4 1 3 3 10 3 1 1 2 6 1 3 4 3 4 4 2 1 3	1 impossible 0

Problem F. Flowers II

In this problem, You will be given a forest. This forest consists of one or multiple directed binary trees. Nodes of the tree can be black or white. You will reconstruct the forest by executing some queries. Note that, each edge directs the relation from parent to child and each node will always have at most one parent. When asked a query, determine the number of **unique flower shapes** in a tree.

Definitions

- **Tree Root:** A node with no parent.
- **Flower Root:** A white node with no parent or a black parent.
- **Flower:** A flower consists of a Flower Root and all the white nodes that can be reached from the flower root via directed edges, with no black nodes appearing along the path.
- **Unique Flower Shapes:** Two Flowers are considered to have the same unique shape if they are structurally identical, meaning:
  - Both flowers have the same number of nodes.
  - The arrangement of left and right children at every node is the same.

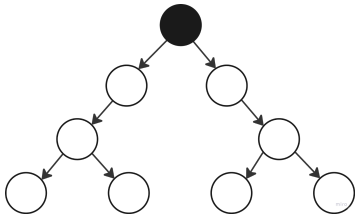


Figure 1: The flowers are in different shapes.

There will be N nodes in the forest. For each node in the forest, you will be given three pieces of information (Left child, Right Child, and Color of that node).

After that, there will be Q instructions. Each of the instructions can be any of the two types below. After each query, the vertices will not break the directed binary tree property. These queries can be any of the following type.

- 1 X Y S:
  This query asks you to merge two trees. Let, R_X = The root of the tree containing the node X. Add R_X as the S (Left or Right) child of Y. If $S = 0$, it represents left, and if $S = 1$, it represents right. It is guaranteed that X and Y are not in the same tree and Y does not have a S child beforehand. Note that, the change persists for the rest of the queries.
- 2 X:
  Determine the number of unique flower shapes in the tree containing the node X.

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 will start with two integers $N(1 \leq N \leq 10^5)$ and $Q(1 \leq Q \leq 3 \times 10^5)$.

The next N lines will contain three space-separated integers L_i, R_i, and C_i ($C_i \in \{0, 1\}$). It is guaranteed that with this information, a valid forest with valid directed binary trees can be formed. L_i or R_i is -1 if the ith node doesn't contain the respective child node. Otherwise, they follow the constraints ($1 \leq L_i, R_i \leq N, L_i \neq R_i$). $C_i = 0$ means the i^{th} node is colored black and $C_i = 1$ denotes the node is colored white.

After building the trees, there will be Q instructions. Each of the instructions can be any of the two types below:

- 1 X Y S
  X, Y and S are integers ($1 \leq X, Y \leq N, S \in \{0, 1\}$).
- 2 X
  X is an integer ($1 \leq X \leq N$).

Note that the sum of N across all test cases is at most 10^5, and the sum of Q across all test cases is at most 3×10^5. It is guaranteed that there will be at least one query of type 2 in each test case.

Output

For each test case, first print `"Case T:"` on a separate line, where T is the test case number. Then, for each query of type 2, print the corresponding answer on a new line.

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

Problem G. Glitch in the Multiverse

The very well-known multiverse has N universes where the i^{th} universe is represented by a n x n square grid, where $(1, 1)$ is the top-left and (n, n) is the bottom-right cell. There are N versions of Silver Man walking on their corresponding grid. The i^{th} version is located at some position (r_i, c_i) on the i^{th} grid, where r_i is the row number and c_i is the column number. From time $t = 0$, each version goes straight in one of the four directions (up, down, left, right), d_i on its grid. They never stop, so they cycle through a row or column depending on their direction. It takes one second to move one cell. So, if a version is at $(1, n)$ and goes right, it will move to $(1, 1)$ after one second. The same applies to walking through a column or in the opposite direction.

Now there is a danger that a glitch will occur in the multiverse if an anomalous situation arises in the neighboring universes. There are Q queries, where an index i, a range $[l, r]$, and a duration s are given in each query. A glitch will occur in the multiverse if the i^{th} version of Silver Man, during its walk for s seconds from $t = 0$, meets more than s distinct versions in total on their walks, where the versions that are met must be in the range $[l, r]$ of the universes. You can say that Silver Man meets a version if and only if they are located in a cell at the same time, or pass each other while walking towards each other in the same row or column. Note that meeting at $t = 0$ is not possible. You have to find out if a glitch will occur for each query.

Input

The first line will contain T $(1 \leq T \leq 10000)$, the number of test cases.

Each case will start with two integers N $(1 \leq N \leq 10000)$, and Q $(1 \leq Q \leq 10000)$.

The second line will contain an integer n $(1 \leq n \leq 2000)$.

Each of the next N lines will contain three integers: r_i $(1 \leq r_i \leq n)$, c_i $(1 \leq c_i \leq n)$, and d_i $(1 \leq d_i \leq 4)$, where, for values 1 through 4, d_i stands for up, down, left, or right, respectively.

Each of the next Q lines will contain four integers: i, l, r $(1 \leq l \leq i \leq r \leq N)$, and s $(1 \leq s \leq n)$.

The summation of N, Q, and n over all test cases does not exceed 10^5, 10^5, and $5 \cdot 10^4$ respectively.

Output

For each query, in a new line, print "YES" if a glitch occurs, otherwise print "NO" (without quotes). **Note: The output is case-sensitive.**

Sample Input	Sample Output
2 4 1 2 1 1 4 2 2 3 2 2 4 2 2 1 4 1 4 2 3 1 3 2 1 4 2 1 4 2 2 3 1 1 3 2	YES NO

Explanation

In the first case, at time t = 1, the 4^{th} version will meet the 1^{st} version at $(1, 2)$. At time t = 2, this version will meet both the 2^{nd} and 3^{rd} version at $(2, 2)$. So, the 4^{th} version will meet three versions in total in these two seconds. As, during the walk, the 4^{th} version meets more than two versions, a glitch will occur in the multiverse.

In the second case, the 1^{st} version will be travelling to $(2, 2)$ and the 3^{rd} version will be travelling to $(2, 1)$ in the first second. The 1^{st} version will meet the 3^{rd} version in the middle of the first second as they pass each other in the opposite direction. At time t = 1, the 1^{st} version will meet the 2^{nd} version at $(2, 2)$. At time t = 2, although all versions will be at $(2, 3)$, it won't add to the count since they would be already met earlier. So, the 1^{st} version will meet exactly two versions in total in these two seconds. As, during the walk, the 1^{st} version does not meet more than two versions, a glitch will not occur.

Problem H. Array Apocalypse

In the post-apocalyptic world, scientists are prohibited from researching. Everyone is forced to work in fields. But Dr. Devil is not like them. He is secretly doing his research on number theory and believes he can change the future of mankind. He has almost completed his research, only thing left is testing the function he calls the GCD Pair Sum(GPS) function.

$$GPS(n) = \sum_{i=1}^{n}\sum_{j=1}^{n} \gcd(i,j)$$

He wants to test GPS on an array. Arrays have evolved, they don't have random access anymore. There is only one function `get()` to access an element of an array. `get()` function does two things:

- Chooses an element from the array randomly and returns it. The probability of choosing i^{th} element $p_i = \frac{A_i}{\sum_{k=1}^{n} A_k}$ (n is current array size)
- Deletes all the multiples of the chosen element from the array.

Note that after deletion, the probability of choosing each element changes. From a secret source, you have learned the array on which Dr. Devil will run his test. He will start calling the `get()` function and calculate the GPS of the obtained number. As well as you know him, he won't stop until the array is fully deleted.

Formally, Dr. Devil starts with an empty set S and inserts `get()` until the array becomes empty. Then he calculates $f(S)$ as the sum of GPS of all elements of S.

As an array lover you can't let him destroy the array. Can you find the expected value of $f(S)$ before Dr. Devil. This is the only way to save the array.

Input

The first line of the input will contain an integer $T(1 \leq T \leq 10)$ denoting the number of test cases. Then T test cases follow. The first line of each test case will contain one integer $N(1 \leq N \leq 10^5)$, length of the array. The second line of each test case will contain N space separated integers $A_1, A_2, \ldots, A_N(1 \leq A_i \leq 10^6)$.
Input file is huge. Use fast I/O.

Output

For each of the test cases, print the expected value of $f(S)$ modulo $10^9 + 7$.

Formally, let $M = 10^9 + 7$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers. Output the integer equal to $p.q^{-1} \bmod M$. In other words, output such an integer x that $0 \leq x < M$ and $x.q \equiv p \pmod{M}$.
Dataset has no such case where q is not co-prime to $M(q \not\equiv 0 \pmod{M})$.

Sample Input	Sample Output
2 2 1 2 2 2 3	333333340 17

Problem I. Let's Learn Multiplication Table

Young LiteratureBoy is trying to learn multiplication tables from his Uncle MathMan, a brilliant mathematician. But LiteratureBoy gets a little confused and writes down the multiplication facts in a jumbled mess!

He writes all the equations correctly, but they may not always follow the usual sequence or consistently maintain the typical placement of the multiplicand and multiplier, as seen in a standard multiplication table. Uncle MathMan is trying to help, but he's having a hard time figuring out which number's multiplication table LiteratureBoy is actually writing.

That's where you come in! Uncle MathMan needs your help to solve this puzzle. He'll show you the jumbled list of equations that LiteratureBoy wrote, and you have to figure out which number's multiplication table he was trying to write.

Input

The first line of the input will contain an integer T ($1 \leq T \leq 10^3$) denoting the number of test cases. Each of the test cases will contain 10 lines, each of which will have the following format: A `x` B `=` C, where $1 \leq A, B \leq 10^9$, C represents the correct product of A and B while `x` is the character with ASCII code 120 (the lowercase letter 'x'). It is guarateed that, either one of the two numbers, A or B will have a value less than or equal to 10.

Output

For each test case, you just need to print the answer.

Sample Input	Sample Output
1 7 x 10 = 70 3 x 10 = 30 9 x 10 = 90 6 x 10 = 60 5 x 10 = 50 4 x 10 = 40 8 x 10 = 80 2 x 10 = 20 10 x 10 = 100 1 x 10 = 10	10

Note

A multiplication table is a list that shows the results of multiplying a specific number by the series of numbers from 1 to 10. For example: the multiplication table of number 1 is:

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

Problem J. T5

Tic-Tac-Toe is a classic game played on a 3×3 grid. Two players take turns marking empty cells with their symbols (commonly 'X' and 'O'). The first player to align three of their symbols in a straight line, either horizontally, vertically, or diagonally, wins. If all cells are filled without a winner, the game ends in a tie. Murad and Himel, having mastered Tic-Tac-Toe, found that their games always ended in a tie.

To make the game more interesting, they invented a new variant **Tic-Tac-Tac-Toe**. In this variant, players can win not only by forming a straight line but also by creating an L-shape in any position or orientation. An L-shape is defined as three symbols forming two perpendicular line segments of length 2, sharing one endpoint. However, as they played more, their skills in this variant also led to frequent ties.

Seeking further challenges, they came up with another version **Tic-Tic-Tac-Tac-Toe**. This version takes the game to a new level by using multiple boards and a unified symbol. Specifically, there are n boards of Tic-Tac-Tac-Toe placed side by side. Murad and Himel alternate turns, with Murad always starting. The rules are as following:

- On each turn, a player selects a board and marks an unmarked cell on that board.
- Unlike the previous games, here both players use *the same symbol*.
- The first player to win on *any one board* by forming a straight line or an L-shape is declared the winner.

This new version is always guaranteed to have a winner and turned out to be highly competitive. In some matches, the game reached a state where both Murad and Himel sought your help. To maintain your reputation as a great programmer (and pro gamer), you will only help someone if you can guarantee their victory no matter what the opponent does.

Given the current state of the game, whose side are you taking?

Input

The first line of the input contains a single positive integer t ($1 \le t \le 5 \times 10^5$) — the number of test cases. Then t test cases follow.

The first line of each test case contains two integers n ($1 \le n \le 5 \times 10^5$) and m ($0 \le m \le 9n$) — the number of boards and the number of moves already played. The next m lines describe the moves made so far.

Each move is described by three integers b_i ($1 \le b_i \le n$), r_i ($1 \le r_i \le 3$) and c_i ($1 \le c_i \le 3$) — the index of the board, row and column respectively where the i-th move ($1 \le i \le m$) was made.

The moves are provided in order. The first move is made by Murad, the second by Himel, and so on.

It is guaranteed that all given moves are valid and the game is still undecided (i.e., no board has a winner yet).

It is also guaranteed that the sum of n over all test cases does not exceed 5×10^5.

Input file is large. Please be sure to use faster input/output methods.

Output

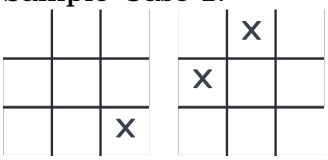
For each test case, output a single line containing either `"Murad"` or `"Himel"` (without the quotes) — the player you can guarantee victory for, regardless of the opponent's strategy. **Note: The output is case-sensitive.**

Sample Input	Sample Output
4 2 3 1 3 3 2 1 2 2 2 1 1 1 1 2 2 5 0 7 1 5 1 3	Himel Murad Murad Himel

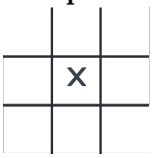
Explanation

The following are the illustrations of the sample test cases:

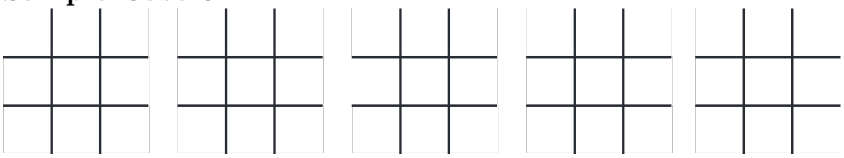
Sample Case 1:



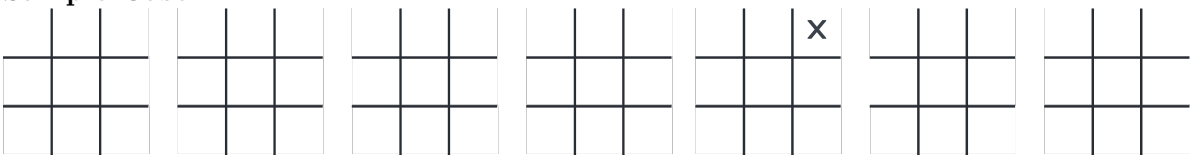
Sample Case 2:



Sample Case 3:



Sample Case 4:



In the first test case, Himel will take the next turn and can win the game right away by creating an L-shape in the second board.

In the second test case, Himel will take the next turn. Whatever he does in that turn, Murad can win the game in the turn after that.

Problem K. Total Distance Over All Trees

Let $d(u, v)$ be the distance between nodes u and v in a tree, defined as the number of edges in the unique path between u and v. For a given positive integer n, consider all possible trees with n vertices such that:

- Each vertex is labelled with a unique integer from 1 to n.
- The trees are non-rooted (no vertex is designated as the root).
- Two trees are considered different if and only if there exist two nodes labelled u and v that are connected by an edge in one tree but not in the other.

For each such tree, calculate $d(1, n)$ - the distance between vertices labelled 1 and n. You need to find the sum of these distances over all possible trees modulo $10^9 + 7$.

Input

The first line contains an integer T ($1 \leq T \leq 10^4$) denoting the number of test cases.

For each test case, there is a single line containing an integer n ($1 \leq n \leq 10^7$) denoting the number of nodes in the tree.

The sum of n across all test cases does not exceed 10^7.

Output

For each of the test cases, print the sum of $d(1, n)$ over all possible labelled non-rooted trees of n vertices modulo $10^9 + 7$.

Sample Input	Sample Output
6 1 2 3 4 5 6	0 1 4 26 236 2760

Explanation

Sample Case 1:
There is only one possible tree with a single vertex labelled 1. Since both vertices 1 and n are the same vertex, $d(1, 1) = 0$. Therefore, the total sum is 0.

Sample Case 2:
There is only one possible labelled non-rooted tree with two vertices - a single edge connecting vertices 1 and 2. In this tree, $d(1, 2) = 1$. Therefore, the total sum is 1.

Sample Case 3:
There are 3 possible labelled non-rooted trees with three vertices.

1. 1-2-3 where vertex 2 connects vertices 1 and 3, giving $d(1,3) = 2$.

2. 1-3-2 where vertex 3 connects vertices 1 and 2, giving $d(1,3) = 1$.

3. 2-1-3 where vertex 1 connects vertices 2 and 3, giving $d(1,3) = 1$.

The total sum is $2 + 1 + 1 = 4$.

Sample Case 4:
There are 16 possible labelled non-rooted trees with four vertices. The sum of $d(1,4)$ over all these trees equals 26.

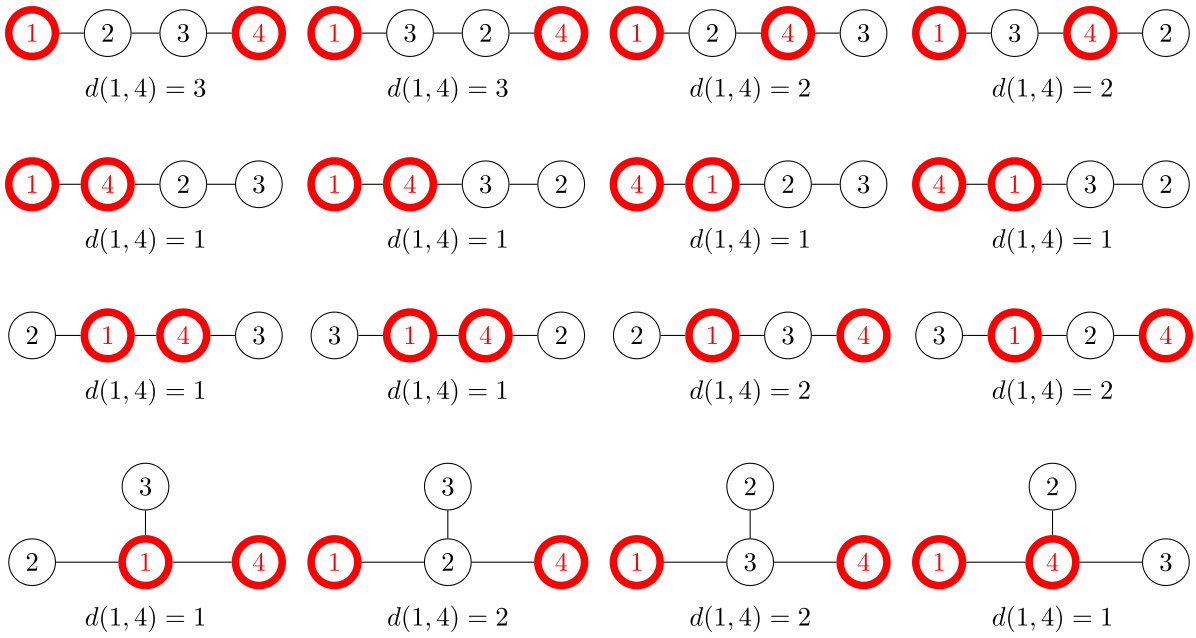


Figure 2: All possible trees for n = 4.