

Problem A. Interval and Expected Value

Time limit: 1 second
Memory limit: 512 megabytes

You are given an integer interval $[l, r]$. At first, your score sco is equal to 0.

Repeat the following process:

1. Choose an integer $x \in [l, r]$ uniformly at random, and update $sco := sco + x$.
2. If $l = r$, stop. Otherwise, replace $[l, r]$ by a uniformly random proper subinterval of $[l, r]$ and go back to step 1. In other words, letting $len = r - l + 1$, there are $\frac{len(len+1)}{2} - 1$ proper subintervals of $[l, r]$, each chosen with equal probability.

Your task is to find the expected value of sco . Output the answer modulo 998 244 353.

Formally, let $M = 998\,244\,353$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \pmod{M}$. In other words, output such an integer x that $0 \leq x < M$ and $x \cdot q \equiv p \pmod{M}$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^5$). The description of the test cases follows.

The only line of each testcase contains two integers l and r ($1 \leq l \leq r \leq 2 \cdot 10^6$).

Output

For each test case, print a single integer — the answer modulo 998 244 353.

Example

standard input	standard output
4	5
2 3	1
1 1	454755778
5 8	881266578
1 2000000	

Note

In the first test case, the initial interval is $[2, 3]$. The following events occur with equal probability:

- sco is increased by 2 $\rightarrow [l, r]$ becomes $[2, 2] \rightarrow sco$ is increased by 2 $\rightarrow$ stop. The final value of sco is 4.
- sco is increased by 3 $\rightarrow [l, r]$ becomes $[3, 3] \rightarrow sco$ is increased by 3 $\rightarrow$ stop. The final value of sco is 5.
- sco is increased by 3 $\rightarrow [l, r]$ becomes $[2, 2] \rightarrow sco$ is increased by 2 $\rightarrow$ stop. The final value of sco is 5.
- sco is increased by 3 $\rightarrow [l, r]$ becomes $[3, 3] \rightarrow sco$ is increased by 3 $\rightarrow$ stop. The final value of sco is 6.

Thus, the expected value of sco is $\frac{4+5+5+6}{4} = 5$.

Problem B. Tree Path Price Queries

Time limit: 2 seconds
Memory limit: 512 megabytes

You are given a tree with n nodes, rooted at node 1. Each node i contains k_i items, all having the same price p_i .

You need to process several queries. Each query is described by four integers: u , d , p_{lower} , and p_{upper} .

For each query, you must determine the number of items that satisfy all of the following conditions:

- Their price lies within the range $[p_{\text{lower}}, p_{\text{upper}}]$.
- Their corresponding node lies within distance d from node u .
- Their node lies on the shortest path from node u to the root node (node 1).

Input

The first line contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of nodes in the tree.

Each of the next $n - 1$ lines contains two integers a and b ($1 \leq a, b \leq n$; $a \neq b$), denoting an edge between nodes a and b .

Then n lines follow. The i -th of these lines contains two integers k_i and p_i ($0 \leq k_i \leq 10^5$, $1 \leq p_i \leq 10^9$) — the number of items and their common price at node i .

Then an integer q ($1 \leq q \leq 2 \cdot 10^5$) is given — the number of queries.

In the i 'th line of the next q lines contains four integers U_i ($1 \leq U_i \leq n$), D_i ($0 \leq D_i \leq n - 1$), PL_i and PU_i ($1 \leq PL_i, PU_i \leq 10^9$).

Let $ans_0 = 0$. For each query i , the actual parameters are decoded as follows:

$$\begin{aligned} u &= ((U_i + ans_{i-1}) \bmod n) + 1, \\ d &= (D_i + ans_{i-1}) \bmod n, \\ p_{\text{lower}} &= ((PL_i + ans_{i-1}) \bmod 10^9) + 1, \\ p_{\text{upper}} &= ((PU_i + ans_{i-1}) \bmod 10^9) + 1. \end{aligned}$$

If $p_{\text{lower}} > p_{\text{upper}}$, then these two values are swapped.

After decoding $1 \leq u \leq n$, $0 \leq d \leq n - 1$, $1 \leq p_{\text{lower}} \leq p_{\text{upper}} \leq 10^9$.

After printing the answer to query i , it becomes ans_i , which is then used in decoding query $i + 1$.

Output

For each query, print a single integer — the number of items satisfying the conditions above.

Example

standard input	standard output
10	20
1 2	0
2 3	0
3 4	0
4 5	8
1 6	
1 7	
3 8	
2 9	
3 10	
3 4	
8 5	
8 8	
3 8	
8 7	
0 8	
0 1	
1 9	
4 5	
4 10	
5	
7 5 1 10	
10 5 7 10	
2 1 10 10	
8 7 5 10	
2 5 7 10	

Note

- Query 1 ($u = 8, d = 5, p_{lower} = 2, p_{upper} = 11$): path $8 \rightarrow 3 \rightarrow 2 \rightarrow 1$, all nodes in the path range $[2, 11]$, total 20 items.
- Query 2 ($u = 1, d = 5, p_{lower} = 28, p_{upper} = 31$): path 1, no node in price range $[28, 31]$, total 0 item.
- Query 3 ($u = 3, d = 1, p_{lower} = 11, p_{upper} = 11$): path $3 \rightarrow 2$, no node with price 11, total 0 item.
- Query 4 ($u = 9, d = 7, p_{lower} = 6, p_{upper} = 11$): path $9 \rightarrow 2 \rightarrow 1$, no node in price range $[6, 11]$, total 0 item.
- Query 5 ($u = 3, d = 5, p_{lower} = 8, p_{upper} = 11$): path $3 \rightarrow 2 \rightarrow 1$, node 3 in price range $[8, 11]$, total 8 items.

Problem C. Max Person

Time limit: 1 second
Memory limit: 256 megabytes

You have an infinite-storey building. Each floor has exactly n rooms, and each room can accommodate one person.

Placing one person on floor x (where floors are 1-indexed: $x = 1, 2, 3, \dots$) costs 2^x units of budget. You are given a total budget of m and you must spend all of it.

Your task is to determine the maximum possible number of people you can accommodate while spending exactly m . If it is impossible to spend the budget exactly, output -1 .

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 ($1 \leq n \leq 10^{18}$) and m ($1 \leq m \leq 10^{18}$).

Output

For each test case, print a single integer on its own line — the maximum number of people that can be accommodated while spending exactly m , or -1 if it is impossible.

Example

standard input	standard output
2	5
5 10	4
1 30	

Note

For the first case: $n = 5, m = 10$. It is optimal to place 5 people on floor 1. Cost is $5 \cdot 2^1 = 10$ — exact, so the answer is 5.

For the second case 2: $n = 1, m = 30$. At most one person can be accommodated per floor. If we place one person on floors 1, 2, 3, 4 (cost $2 + 4 + 8 + 16 = 30$). Total people = 4, which is maximal since each floor can host only one person.

Problem D. The New CEO of CloseAI

Time limit: 4 seconds
Memory limit: 256 megabytes

Fahim had only been at CloseAI for two weeks, but he already knew two things with absolute certainty:

1. The company's tokenization system was held together with duct tape, prayers, and legacy Bash scripts.
2. He did not like the CEO, Saltman, who walked around the office wearing sunglasses indoors and calling everyone "chat-bros".

One morning, Saltman burst into the engineering floor holding a pile of printed stack traces like a kindergarten art project gone wrong.

"Tokenization is broken again!" he shouted. "If we don't fix this, our shareholders will replace me with a bag of potatoes!"

Fahim stared at him. A thought crossed his mind: If I fix this, maybe they'll replace him with me.

The challenge? Every model at CloseAI was panicking because they had forgotten how to tokenize words. They needed a clean implementation of tokenization, or as Saltman called it, "that spicy token cruncher thing".

So Fahim sat down, cracked his knuckles, and dove into the problem:

You're given n lowercase English words, and you must apply tokenization to them, like a hero saving a collapsing startup. Each unique character from 'a' to 'z' starts as its own token — basic, simple, unlike Saltman's understanding of machine learning.

Then comes the tokenization loop:

Fahim has to count all adjacent token pairs across all words. He must find the most frequent adjacent token—the one that appears more than anything else. If there's a tie, he chooses the lexicographically smallest adjacent token. Then he merges that pair into a new token, like smashing two puzzle pieces together with pure engineering rage. Fahim also has to handle overlapping cases properly. For example, in a word like 'aaa', the pair 'aa' appears twice but overlaps — he only merges the leftmost occurrence first, so 'aaa' becomes ['aa', 'a'] in one merge step.

He repeats this: count, pick, merge, replace everywhere. He keeps going until there are no adjacent pairs of tokens. That's when the algorithm ends, and all the final tokens must be printed.

As he coded, something incredible happened: the errors stopped. The server logs went silent. The models started tokenizing again without screaming across the datacenter. Before anyone realized, the entire company was running smoothly.

Saltman strutted over and said, "Nice work, chat-bro. You've saved CloseAI."

Fahim smiled. One step closer to taking your job, Saltman.

Input

A single line containing a string s — one or more words separated by single spaces.

Each word is a valid lowercase English **dictionary** word consisting only of characters from **a** to **z**.

It is guaranteed that each word has length at most 10 and the total number of words does not exceed 10^5 .

Output

Print each unique token on a separate line in lexicographically sorted order.

Example

standard input	standard output
cat bat rat	a at b bat c cat r rat t

Note

In the first example:

Initial state:

Each character is a token: c, a, t, b, r.

Words: cat $\rightarrow$ [c, a, t], bat $\rightarrow$ [b, a, t], rat $\rightarrow$ [r, a, t]

Step 1: Count adjacent pairs

- 1st: [c, a, t]: (c, a) $\rightarrow$ "ca", (a, t) $\rightarrow$ "at"
- 2nd: [b, a, t]: (b, a) $\rightarrow$ "ba", (a, t) $\rightarrow$ "at"
- 3rd: [r, a, t]: (r, a) $\rightarrow$ "ra", (a, t) $\rightarrow$ "at"

Frequencies: "at" appears 3 times, "ca", "ba", "ra" each appear 1 time.

Most frequent pair: "at" $\rightarrow$ merge (a, t) into token at.

After merge:

cat $\rightarrow$ [c, at], bat $\rightarrow$ [b, at], rat $\rightarrow$ [r, at]

Step 2: Count adjacent pairs

- 1st: [c, at]: (c, at) $\rightarrow$ "cat"
- 2nd: [b, at]: (b, at) $\rightarrow$ "bat"
- 3rd: [r, at]: (r, at) $\rightarrow$ "rat"

Frequencies: "cat", "bat", "rat" each appear 1 time (tie).

Lexicographically smallest: "bat" < "cat" < "rat" $\rightarrow$ merge (b, at) into token bat.

After merge:

cat $\rightarrow$ [c, at], bat $\rightarrow$ [bat], rat $\rightarrow$ [r, at]

Step 3: Count adjacent pairs

- 1st: [c, at]: (c, at) $\rightarrow$ "cat"
- 3rd: [r, at]: (r, at) $\rightarrow$ "rat"

Frequencies: "cat", "rat" each appear 1 time (tie).

Lexicographically smallest: "cat" < "rat" → merge (c, at) into token cat.

After merge:

cat → [cat], bat → [bat], rat → [r, at]

Step 4: Count adjacent pairs

- 3rd: [r, at]: (r, at) → "rat"

Only one pair: "rat" → merge (r, at) into token rat.

Final state:

cat → [cat], bat → [bat], rat → [rat]

All unique tokens: a, at, b, bat, c, cat, r, rat, t

Output in lexicographically sorted order.

Problem E. Toggle the Streetlights

Time limit: 1 second
Memory limit: 256 megabytes

The Metropolitan University campus has a long street with n streetlights arranged in a row, numbered from 1 to n . Each streetlight is either on (represented by 1) or off (represented by 0).

The streetlights have a peculiar automatic control system. Every minute, the system updates all streetlights simultaneously according to the following rule:

For a streetlight at position i (where $1 \leq i \leq n$), we define its **neighbors** as all positions j such that $|i - j| = 1$ and $1 \leq j \leq n$.

A streetlight at position i (where $1 \leq i \leq n$) toggles its state (on $\rightarrow$ off or off $\rightarrow$ on) if and only if it has **exactly two neighbors** and **both of them were on** in the previous minute.

You are given the initial state of all streetlights. Your task is to determine the state of all streetlights after exactly k minutes.

Input

The first line contains a single integer T ($1 \leq T \leq 10^5$) — the number of test cases.

Each test case consists of two lines:

- The first line contains two integers n and k ($1 \leq n \leq 10^5$, $0 \leq k \leq 10^9$) — the number of streetlights and the number of minutes, respectively.
- The second line contains a binary string s of length n — the initial state of the streetlights, where $s_i = 1$ means the i -th streetlight is on, and $s_i = 0$ means it is off.

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

Output

For each test case, print a single line containing a binary string of length n — the state of the streetlights after k minutes.

Example

standard input	standard output
1 5 1 01010	01110

Problem F. Time Well Spent

Time limit: 1 second
Memory limit: 256 megabytes

Alice and Bob have been close ever since high school. Now they both study at Martian University (MU), but in different departments located far apart. Their schedules are packed, and their classes take place in buildings scattered across the campus. Even though they want to spend time together, they can only meet during the short breaks between their classes.

The campus has N buildings connected by M undirected roads. Each road requires a certain amount of time to walk. Both Alice and Bob have several classes throughout the day, each class occurs at a specific building and lasts between a given start and end time. Class intervals do not overlap for each student. Since days on Mars do not follow Earth's fixed length, MU measures time on its own continuous scale, and a single day may be arbitrarily long. As a result, class start and end times can be large values.

Between two consecutive classes of their own, a student may roam the campus freely, traveling along any roads and visiting any buildings, as long as they arrive at their next class on time. Alice and Bob can meet only if they are in the same building at the same moment, during free periods for both. They are dedicated to their studies and will not be late to any class. They also work part time before and after their class schedules, so they will only meet between classes, never before the first class or after the last one.

Alice and Bob may choose different buildings to meet during different free intervals. Your task is to determine the maximum total time they can spend together during the day.

Input

The first line contains one integer t ($1 \leq t \leq 10$), the number of test cases.

Each test case begins with a line containing two integers N and M ($1 \leq N \leq 200$, $1 \leq M \leq \frac{N(N-1)}{2}$), where N is the number of buildings and M is the number of roads. Each of the next M lines contains three integers u, v, w ($1 \leq u, v \leq N$, $1 \leq w \leq 10^5$), describing a bidirectional road between buildings u and v with travel time w minutes.

Then a line follows containing two integers A and B , the number of classes Alice and Bob have, respectively. The next A lines describe Alice's classes; each line contains three integers $start, end, loc$ ($1 \leq start < end \leq 10^{15}$, $1 \leq loc \leq N$). After that, the next B lines describe Bob's classes in the same format.

The sum of $A + B$ over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — the maximum total time Alice and Bob can spend together.

Example

standard input	standard output
1 4 5 1 2 3 1 3 4 2 3 2 3 4 1 2 4 1 2 2 1 10 1 20 30 2 1 10 1 20 30 3	6

Problem G. Awkward Nodes

Time limit: 1 second
Memory limit: 256 megabytes

You are given a tree consisting of N nodes numbered from 1 to N , connected by $N - 1$ edges.
You are also given a list of M distinct “special” nodes. The remaining $N - M$ nodes are “normal.”
You can move between nodes along the tree edges.

- Normal nodes can be visited any number of times.
- Each special node can be visited at most once during your entire journey.

For each node i ($1 \leq i \leq N$), determine the **maximum number of distinct nodes** that can be visited in a single continuous journey starting from node i .

Input

The first line contains a single integer T ($1 \leq T \leq 2 \cdot 10^5$), the number of test cases. The description of each test case follows:

- The first line contains two integers N and M ($1 \leq M \leq N \leq 2 \cdot 10^5$).
- The second line contains M distinct integers $s_1, \dots, s_M$ ($1 \leq s_i \leq N$), the special nodes.
- The next $N - 1$ lines each contain two integers u and v ($1 \leq u, v \leq N$), denoting an edge.

It is guaranteed that the sum of N over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print N space-separated integers, where the i -th integer is the answer for a starting journey from node i .

Example

standard input	standard output
2	3 2 3
3 1	2 3 3
2	
1 2	
2 3	
3 1	
1	
1 2	
1 3	

Note

Test Case 1:

- Start at 1: The journey $1 \rightarrow 2 \rightarrow 3$ is valid. Maximum number of unique nodes visited: 3.
- Start at 2 (Special): The journey can be $2 \rightarrow 1$ or $2 \rightarrow 3$. A journey like $2 \rightarrow 1 \rightarrow 2 \rightarrow 3$ is invalid as it requires revisiting node 2. Maximum number of unique nodes visited: 2.
- Start at 3 (Normal): The journey $3 \rightarrow 2 \rightarrow 1$ is valid. Maximum number of unique nodes visited: 3.

Problem H. Guess the Number

Time limit: 1 second
Memory limit: 256 megabytes

*This is an interactive problem. You **must** flush the output after printing each line. For example, in C++ use `fflush(stdout)`, in Java — `System.out.flush()`, in Pascal — `flush(output)`, and in Python — `sys.stdout.flush()`.*

Two players, A and B, are playing on a pile of N stones. Player A moves first, and then they alternate turns. On each move, a player may remove any number of stones from 1 to k , but not more than the number currently in the pile. A player who cannot move loses.

The value of k is an unknown integer satisfying $1 \leq k \leq 1000$. Your task is to determine this unknown value.

You may ask queries. A query is a single integer N ($1 \leq N \leq 10^{18}$). For this N , the system simulates optimal play and replies with “A” if player A wins, or “B” if player B wins.

You may ask **at most** 70 queries (not counting the final answer).

Input

The first line contains a single integer T ($1 \leq T \leq 100$) — the number of test cases.

Following lines will contain responses to your queries — “A” or “B”. The i -th line is a response to your i -th query.

You can read a response **only after** printing a query and flushing the output.

Output

To ask a query, output “? n ” (without quotes), where $1 \leq n \leq 10^{18}$, followed by a flush operation.

When you determine the unknown value k , output “! k ” (without quotes), where $1 \leq k \leq 1000$, flush the output, and continue to the next test case.

Example

standard input	standard output
1	? 1
A	? 765
B	? 1000
A	! 50

Problem I. Fruit Ninja

Time limit: 1 second
Memory limit: 256 megabytes

You are playing a 3D version of **Fruit Ninja**. A strange fruit appears — it is shaped like a **tetrahedron**, defined by 4 points in 3d-space that do not lie on the same plane.

To get a perfect score, you must slice the fruit with a single **plane** so that the two pieces have **exactly equal volume**.

The fruit's vertices are:

$$P_1(x_1, y_1, z_1), P_2(x_2, y_2, z_2), P_3(x_3, y_3, z_3), P_4(x_4, y_4, z_4)$$

Your task is to output **any plane** that cuts the tetrahedron into two equal-volume parts.

A plane is written as: $ax + by + cz + d = 0$

A correct answer is guaranteed to exist, and if there are multiple such slicing planes, you may output any.

Input

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

For each test case, the input contains four lines. Each line has three integers x_i, y_i, z_i ($|x_i|, |y_i|, |z_i| \leq 10^4$) — the coordinates of the tetrahedron's vertices (P_1, P_2, P_3, P_4).

Output

For each test case, print four real numbers a, b, c, d — defining a plane ($ax + by + cz + d = 0$) that splits the tetrahedron into two equal volumes. Any of $|a|, |b|, |c|, |d|$ are bounded by 10^9 .

Your answer is accepted if the absolute or relative error is less than or equal to 10^{-6} .

Example

standard input	standard output
2	0.47778 -3.00000 -2.40000 4.00000
0 0 0	-11.01667 -2.88333 -0.19444 31.45000
0 0 2	
0 2 0	
2 1 1	
1 2 6	
2 4 3	
2 5 9	
1 7 9	

Problem J. Insert Force

Time limit: 1 second
Memory limit: 256 megabytes

You are given an array a of size n consisting of non-negative integers. At first, your score is 0.

You can perform the following operation:

- Choose an index i ($1 \leq i < |a|$), increase your score by $(a_i + a_{i+1})$.
- Then, insert $(a_i + a_{i+1})$ between a_i and a_{i+1} . Note that after inserting, the size of the array is increased by 1.

You need to answer q **independent** queries. For the i -th query, given a number x_i , you should find the maximum score you can obtain after performing **exactly** x_i operations. Since the answer may be large, output the answer modulo 998 244 353.

Input

The first line contains an integer t ($1 \leq t \leq 10^5$), the number of test cases.

For each test case, the input consists of four lines:

- The first line contains an integer n ($2 \leq n \leq 2 \cdot 10^5$), representing the size of the array.
- The second line contains n space-separated integers $a_1, a_2, \dots, a_n$ ($0 \leq a_i \leq 2 \cdot 10^5$).
- The third line contains an integer q ($1 \leq q \leq 2 \cdot 10^5$), representing the number of queries.
- The fourth line contains q space-separated integers $x_1, x_2, \dots, x_q$ ($1 \leq x_i \leq 2 \cdot 10^5$).

It is guaranteed that the sum of n and q over all test cases will not exceed $2 \cdot 10^5$.

Output

For each test case, output q space-separated integers in a single line, representing the maximum score after performing x_i operations, modulo 998 244 353.

Example

standard input	standard output
2	15 40
4	258200160 1100098 60800104 385867226
8 7 2 12	
2	
1 2	
6	
100000 0 99882 191 99699 487	
4	
15 4 12 200000	

Note

In the first test case,

- If you do exactly 1 operation, the optimal scheme is $[8, 7, 2, 12] \rightarrow [8, \underline{15}, 7, 2, 12]$.
- If you do exactly 2 operations, the optimal scheme is $[8, 7, 2, 12] \rightarrow [8, 7, 2, \underline{14}, 12] \rightarrow [8, 7, 2, 14, \underline{26}, 12]$.