SMU ICPC Selection Contest 2026

(Problemset v6.4)

18 July 2026
Duration: 5 hours

SMU Competitive Programming

Problem Author: Jonathan Teo

Blank lines are provided in sample inputs to separate test cases clearly. They are omitted in the actual input.
Problems are divided into **three groups**, but the problems within each group are not ordered by difficulty.

Group 1 (CF 900–1200 / AtCoder 200–600) Problems A, B
Group 2 (CF 1400–1800 / AtCoder 900–1400) Problems C, D, E, F, G
Group 3 (CF 1800–2500 / AtCoder 1400–2300) Problems H, I, J, K

Group 1

This group consists of Problems A and B.

Problem A. Arrival Signs

Time limit: 4 s *Memory limit: 512 MB*

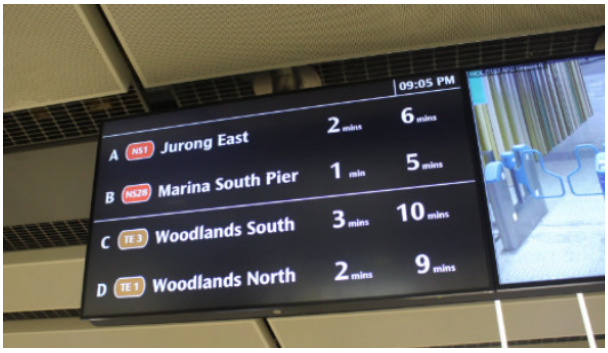

Figure 1: SMRT Information Sign

The MRT Information System tells commuters when the next train will arrive. However, Bob is worried that displaying the next train's arrival time too directly may encourage commuters to run towards the platform, which is dangerous.

Bob is a software developer working on the next iteration of these information signs with safety in mind.

A day starts at time 0. There are n platforms. For platform j, trains arrive every t_j minutes; that is, trains serving platform j arrive at times

$$0, t_j, 2t_j, 3t_j, \cdots$$

The station also has m information signs. A commuter is estimated to take $d_{i,j}$ minutes to walk from sign i to platform j.

Suppose a commuter leaves sign i at time x and walks to platform j. The sign displays the number of minutes from time x until the earliest train they can catch. Formally, a commuter who starts walking from sign i reaches platform j at time

$$x + d_{i,j}.$$

The sign should display the difference between time x and the arrival time of the earliest train at platform j that arrives at or after time $x + d_{i,j}$.

For example, suppose it takes $d_{1,1} = 5$ minutes to walk from sign 1 to platform 1, and trains arrive at platform 1 every $t_1 = 6$ minutes.

- At time $x = 1$, the commuter reaches the platform at time 6. The earliest train they can catch arrives at time 6, so the sign displays $6 - 1 = 5$.
- At time $x = 2$, the commuter reaches the platform at time 7. The earliest train they can catch arrives at time 12, so the sign displays $12 - 2 = 10$.

Help Bob answer q queries. For each query (x, i, j), output the number of minutes displayed on sign i for platform j at time x minutes from the start of the day.

The input, output specifications and sample test cases are found on the next page.

Input

- The first line contains 2 integers n and m ($1 \leq n, m \leq 100$), the number of platforms n and number of signs m.
- The second line contains n integers $t_1, t_2, \cdots, t_n$ ($1 \leq t_j \leq 400$), the j-th integer denoting the interval between consecutive train arrivals at platform j.
- The next m lines contain n integers each. The i-th line contains the n integers $d_{i,1}, d_{i,2}, \cdots, d_{i,n}$ ($1 \leq d_{i,j} \leq 400$). The j-th integer on the i-th line, $d_{i,j}$, denotes the time taken to walk from sign i to platform j.
- The next line contains a single integer q ($1 \leq q \leq 2 \times 10^5$), the number of queries.
- The next q lines each contain three integers x, i, j ($0 \leq x \leq 10^3$, $1 \leq i \leq m$, $1 \leq j \leq n$), the k-th line containing the values (x, i, j) for the k-th query.

Output q lines, where the i-th line ($1 \leq i \leq q$) contains the answer to the i-th query.

Sample Input	Sample Output
3 2 5 7 1 1 2 3 5 7 10 8 0 1 1 4 1 1 5 1 1 6 1 2 7 2 2 1000 1 3 0 2 1 3 2 1	5 1 5 8 7 3 5 7

Problem B. Billy Loves Zeros

Time limit: 2 s *Memory limit: 512 MB*

Once upon a time, Billy had an array a of length n such that $a_i = i$ for all $1 \leq i \leq n$, and an integer m ($m > n$). The array a was then cyclically shifted to the right by k positions to form the array b.

For example, when $n = 5$ and $k = 2$, the arrays a and b are given as follows:

$$a = [1, 2, 3, 4, 5] \longrightarrow \text{cyclic right-shift by } 2 \longrightarrow b = [4, 5, 1, 2, 3]$$

Billy may perform one type of operation on array b, consisting of two steps:

1. Choose two integers l and r such that $1 \leq l \leq r \leq n$
2. For each index $i \in [l, r]$, set $b_i \leftarrow (b_i + 1) \bmod m$

Billy especially loves arrays in which every element is 0. Help Billy find the minimum number of operations required to transform b into $[0, 0, \cdots, 0]$.

Input The first line contains an integer t ($1 \leq t \leq 2 \times 10^5$), the number of test cases.

Each test case consists of a single line containing three integers n, k, m ($1 \leq n < m \leq 10^9$, $0 \leq k \leq n - 1$), denoting the size of the array b, the right-shift offset k and the modulus m.

Output t lines, the answer to each test case on a single line.

Sample Input	Sample Output
2 1 0 3 5 2 6	2 6

Note There are 2 test cases.

In the first test case, $a = [1]$ and since $k = 0$, $b = a = [1]$. Let $f(l, r)$ denote one application of the operation to the subarray from index l to index r:

- Operation 1 - $f(1, 1) = [(1 + 1) \bmod 3] = [2]$
- Operation 2 - $f(1, 1) = [(2 + 1) \bmod 3] = [0]$

After these two operations, $b = [0]$. Therefore the minimum number of operations is 2.

Group 2

This group consists of Problems C, D, E, F and G.
These problems are relatively challenging.
Do keep an eye on the scoreboard and remember that difficulty is subjective.
If you feel stuck, read the other problems in this group.

Problem C. Can You BELIEVE It?

Time limit: 2 s *Memory limit: 512 MB*

At Britain's Got Talent 2026, Rafferty wants to perform a card trick using a deck of n cards in a predetermined order.

Each card has:

- a **face-up** number from 1 to 13,
- a **face-down** handwritten letter, which is one of `B`, `E`, `L`, `I`, `V`.

The trick begins with all cards in the deck **face-up**.

1. The audience member chooses an index i ($1 \le i \le n - 6$), discards the first $i - 1$ cards, and keeps the next 7 cards, namely positions $i, i + 1, \ldots, i + 6$.
2. These 7 cards are then arranged in increasing order of their face-up numbers.

After that, Rafferty turns them over. The letters on their face-down sides must read

`BELIEVE`

from left to right.

Figure 2: Final result of the trick[2]. The face-down letters of the cards at positions i to $i + 6$ spell `BELIEVE` when arranged in ascending order of face-up numbers. The BGT crowd goes wild — somehow, this is inspirational.

In other words, for the chosen window of 7 cards:

- after sorting the cards by face-up number in increasing order,
- the corresponding face-down letters must be `B`, `E`, `L`, `I`, `E`, `V`, `E`.
- all face-up numbers must be distinct. Otherwise, cards with equal numbers could be reordered arbitrarily, which may destroy the word `BELIEVE`.

Your task is to construct the entire deck so that **this works for every valid choice of** i ($1 \le i \le n - 6$).

Please turn over for the input and output specifications.

[2]Watch the trick at `https://www.youtube.com/watch?v=BCHkMexXu40&t=6m14s`.

Input The input consists of a single line containing an integer n ($n = 7$ or $n = 52$) - the number of cards in the deck. **There are exactly 2 tests for this problem.** The sample has $n = 7$ and one hidden test case has $n = 52$.

Output 2 lines:

- On the first line, output n space-separated integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq 13$), where c_i is the face-up number of the i-th card.
- On the second line, output n space-separated uppercase characters, where the i-th letter is the face-down letter of the i-th card.

Your output must satisfy all of the following:

- Each face-down letter must be one of `B`, `E`, `L`, `I`, `V`.
- Each face-up number from 1 to 13 appears at most 4 times.
- In every contiguous segment of 7 cards, all face-up numbers are distinct.
- For every i with $1 \leq i \leq n-6$, when the cards at positions i to $i+6$ are sorted by face-up number, their face-down letters form `BELIEVE`.

> The constraints are chosen to resemble a standard deck of cards. Ignoring jokers, a standard deck has 4 suits, and each suit contains 13 ranks: Ace, ranks 2 to 10, Jack, Queen and King. In this problem, we represent these 13 ranks using the **face-up numbers** from 1 to 13. Hence, there are 52 cards in total, and each face-up number from 1 to 13 may appear at most 4 times.

Sample Input	Sample Output
`7`	`7 5 4 3 2 1 6` `E E I L E B V`

Note The sample is only meant to illustrate the output format.
For $n = 7$, the only valid choice is $i = 1$. Sorting the 7 cards by face-up number gives the numbers

$$1, 2, 3, 4, 5, 6, 7$$

with corresponding letters

$$\texttt{B}, \texttt{E}, \texttt{L}, \texttt{I}, \texttt{E}, \texttt{V}, \texttt{E}$$

so the word `BELIEVE` is formed.
Since there are only two possible inputs, you may output this exact construction for $n = 7$ and focus on your construction for $n = 52$.

Problem D. Duel of Ten

Time limit: 2 s *Memory limit: 512 MB*

Alice and Bob play a *Duel of Ten* on an array a of length n ($\mathbf{-10^9} \le a_i \le 10^9$). They take turns making moves, with Alice going first. The rules of "Duel-of-Ten" are as follows:

- Let the current length of the array a be m (initially $m = n$).
- On Alice's turn, Alice chooses two distinct indices x and y ($1 \le x < y \le m$) and removes elements a_x and a_y from a, then inserts $\min(a_x, a_y)$ into a.
- On Bob's turn, Bob chooses two distinct indices x and y ($1 \le x < y \le m$) and removes elements a_x and a_y from a, then inserts $\max(a_x, a_y)$ into a.

Thus, each move decreases the length of the array by 1.

The game ends when the array a contains only one element. If the remaining element has an **absolute value less than 10**, Alice wins. Otherwise, Bob wins.

Given an array a, determine who wins if both players play optimally.

Input The first line contains a single integer t ($1 \le t \le 10^3$) - the number of test cases. For each test case,

- The first line contains a single integer n ($2 \le n \le 2 \times 10^5$) - the length of the array a.
- The second line contains n integers $a_1, a_2, \cdots, a_n$ ($\mathbf{-10^9} \le a_i \le 10^9$).

The sum of n over all test cases will not exceed 2×10^5.

Output t lines. For each test case, output `Alice` if Alice wins, and Bob otherwise, assuming both players play optimally.

Please turn over for the sample test cases.

Sample Input	Sample Output
2 4 1 10 1 10 4 1 -10 1 -10	Alice Bob

Note There are 2 test cases. For each test case, we show one possible sequence of moves that ends with the stated winner. These sequences are included only to illustrate the rules; the players do not necessarily play optimally.

- For the first test case, one possible sequence of moves that leads to Alice winning is as follows:
  - The array is initially $a = [1, 10, 1, 10]$.
  - Alice removes 1 and 10, and inserts $\min(1, 10) = 1$ into a. The new array a is $[1, 1, 10]$.
  - Bob removes 1 and 10, and inserts $\max(1, 10) = 10$ into a. The new array a is $[1, 10]$.
  - Alice then chooses 1, 10, and inserts $\min(1, 10) = 1$ into a. The new array a is $[1]$.

  Since $|1| < 10$, Alice wins.
- For the second test case, one possible sequence of moves that leads to Bob winning is as follows:
  - The array is initially $a = [1, -10, 1, -10]$.
  - Alice removes 1 and -10, and inserts $\min(1, -10) = -10$ into a. The new array a is $[-10, 1, -10]$.
  - Bob removes 1 and -10, and inserts $\max(1, -10) = 1$ into a. The new array a is $[1, -10]$.
  - Alice removes 1 and -10 and inserts $\min(1, -10) = -10$ into a. The new array a is $[-10]$.

  Since $|-10| = 10$, Bob wins.

Problem E. Expected Walk Maximization

Time limit: 2 s *Memory limit: 512 MB*

Alice initially has a weighted undirected tree T of n nodes and $n-1$ edges. Each edge has a positive integer weight.

Alice wants to design a route so that she can walk for as long as possible, but she does not want to walk forever. To do so, she chooses a direction for every edge of the tree. In other words, for each undirected edge (u, v), she replaces it with **exactly one** of the directed edges $u \to v$ or $v \to u$, preserving its weight w_{uv}. This produces a directed graph T'.

After constructing T', Alice chooses a starting vertex r ($1 \le r \le n$), and performs the following walk:

- Suppose Alice is currently at vertex u.
- If u has no outgoing edge in T', her walk ends.
- Otherwise, she chooses one outgoing neighbour v **uniformly at random** from all vertices v such that $u \to v$ in T', and traverses that edge.

Traversing an edge of weight w adds w to the total distance walked.

Over all possible choices of edge directions and the starting vertex r, help Alice maximize the **expected total distance** she walks, modulo $10^9 + 7$.

It can be shown that the answer can always be written as a fraction $\frac{P}{Q}$, where $Q \not\equiv 0 \pmod{10^9+7}$. Therefore, output the value

$$P \cdot Q^{-1} \bmod (10^9 + 7)$$

Input The first line of the input contains an integer t ($1 \le t \le 10^3$), the number of test cases. For each test case,

- The first line contains an integer n ($2 \le n \le 10^5$) - the number of nodes.
- The next $n-1$ lines contain three integers u, v, w_{uv} ($1 \le u, v \le n$, $u \ne v$, $1 \le w_{uv} \le 10^9$).

Additional constraints on the input:

- The sum of n over all test cases will not exceed 2×10^5.
- There are no duplicate edges, and the edges are guaranteed to form a tree.

Output For each test case, output a single integer on a line, denoting the maximum **expected total distance** she walks, modulo $10^9 + 7$. Note that the answer should be maximized before taking modulo $10^9 + 7$.

Please turn over for the sample test cases.

Sample Input	Sample Output
1 9 2 1 3 7 9 5 8 7 5 7 3 1 3 4 1 4 5 3 5 6 2 3 2 3	12

Note There is 1 test case. One possible edge orientation is shown in Figure 3.

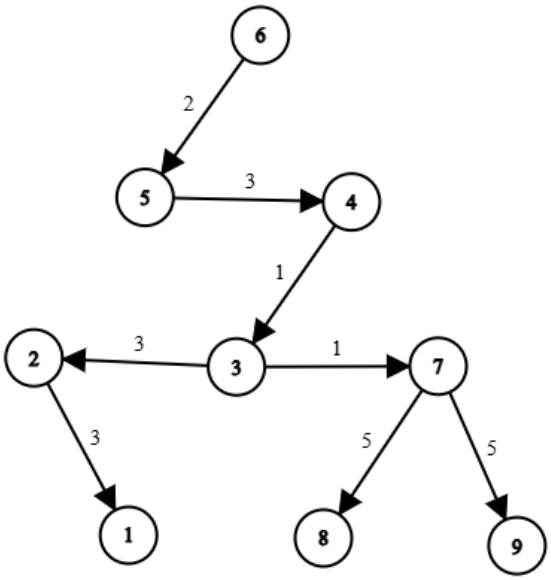

Figure 3: Illustration of Sample Test Case. Visualization created using CSAcademy's graph editor.

When Alice starts from $r = 6$, she is forced to traverse $6 \to 5 \to 4 \to 3$, contributing $2+3+1 = 6$ to her expected distance.
From node 3, she chooses node 2 or node 7 with probability $\frac{1}{2}$ each.

- If she goes to node 2, she is then forced to continue to node 1, adding $3 + 3 = 6$
- If she goes to node 7, she then chooses node 8 or node 9 uniformly, so the expected distance is $1 + \frac{1}{2} \cdot 5 + \frac{1}{2} \cdot 5 = 6$

Therefore, the expected total distance is

$$6 + \frac{1}{2} \cdot 6 + \frac{1}{2} \cdot 6 = 12$$

which can be shown to be the maximum expected total distance of her walk.

Problem F. Flawed Prize Redemption

Time limit: 2 s *Memory limit: 512 MB*

Jon has n tasks and m units of time. There are also m cash prizes, where prize j is worth v_j dollars.

Task i requires a duration of a_i to complete. If Jon completes task i, he becomes eligible for cash prize p_i where

$$p_i = \lfloor \frac{m}{a_i} \rfloor,$$

receiving v_{p_i} dollars. If several completed tasks correspond to the same prize, Jon receives that prize only once.

Jon may choose any subset of tasks and complete them in any order, as long as the total time taken by the chosen tasks does not exceed m.

Find the maximum amount of money, in dollars, that Jon can receive today.

Input The first line contains a single integer t ($1 \le t \le 300$), the number of test cases. For each test case,

- The first line contains two integers n and m ($1 \le n, m \le 10^5$), the number of tasks n and available units of time m. There are also m cash prizes.
- The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le m$), where the i-th integer is the duration of the i-th task.
- The third line contains m integers $v_1, v_2, \cdots, v_m$ ($1 \le v_j \le 10^9$), where the j-th integer is the cash prize amount of the j-th cash prize, in dollars.

Additional constraints of input:

- The sum of n over all test cases will not exceed 10^5.
- The sum of m over all test cases will not exceed 10^5.

Output t lines. For each test case, output a single integer: the maximum amount of money, in dollars, that Jon can receive today.

Please turn over for the sample test cases.

Sample Input	Sample Output
2 6 10 1 2 3 4 6 10 1 5 20 1 30 1 1 1 1 60 7 12 2 3 4 5 6 11 12 1 8 18 35 1 40 1 1 1 1 1 1	115 93

Note There are two test cases.

- For the first test case, $m = 10$. The list below gives task durations and corresponding prize indices:
  - $a_1 = 1$, $p_1 = \lfloor 10/1 \rfloor = 10$
  - $a_2 = 2$, $p_2 = \lfloor 10/2 \rfloor = 5$
  - $a_3 = 3$, $p_3 = \lfloor 10/3 \rfloor = 3$
  - $a_4 = 4$, $p_4 = \lfloor 10/4 \rfloor = 2$
  - $a_5 = 6$, $p_5 = \lfloor 10/6 \rfloor = 1$
  - $a_6 = 10$, $p_6 = \lfloor 10/10 \rfloor = 1$

  One optimal choice is to complete tasks 1, 2, 3, 4, and the total time is $1+2+3+4 = 10$. The total amount of money he receives is $60 + 30 + 20 + 5 = 115$ dollars.
- In the second test case, one optimal choice is to complete tasks 1, 2 and 3. The total time used is $2 + 3 + 4 = 9$. The cash prizes received are $40 + 35 + 18 = 93$.

Problem G. Google Maps Does Not Work Here

Time limit: 2 s *Memory limit: 512 MB*

You live in a city with a mysterious transportation network consisting of n locations and m bi-directional roads. For the i-th road ($1 \leq i \leq m$), the time taken to travel across it is 2^{i-1} minutes.

Alice is new to the city and is getting ready for her road trips. However, to her horror, Google Maps is not available in this city!

Alice turns to you, an aspiring competitive programmer. Alice's house is located at location 1. For every location v ($2 \leq v \leq n$), please help Alice find the minimum travel time from location 1 to location v, and output that time modulo 998244353.

Input The first line contains a single integer t ($1 \leq t \leq 10^3$), the number of test cases. For each test case,

- The first line consists of two integers n and m ($2 \leq n \leq 2 \times 10^5$, $1 \leq m \leq 5 \times 10^5$), denoting the number of nodes n, the number of bi-directional roads m.
- The following m lines contain two integers u and v ($1 \leq u, v \leq n$, $u \neq v$), denoting a bi-directional road between locations u and v. The i-th road listed in the input takes 2^{i-1} minutes to travel across.

Additional constraints on input:

- The sum of n over all test cases will not exceed 2×10^5.
- The sum of m over all test cases will not exceed 5×10^5.
- The transportation network is a simple graph, with no multiple edges or self-loops.

Output t lines. For each test case, output one line containing $n-1$ integers $d_2, d_3, \cdots, d_n$, where d_i is defined by:

- If at least one path exists from location 1 to location i, then d_i should be the minimum possible travel time from location 1 to location i, modulo 998244353. Note that the travel time itself should be minimized before taking the result modulo 998244353.
- Otherwise, d_i should be `-1`.

Please turn over for the sample test case.

Sample Input	Sample Output
1 6 7 2 3 3 4 2 4 4 5 2 5 1 5 1 2	43 42 40 32 -1

Note There is 1 test case. Figure 4 provides a visualisation.

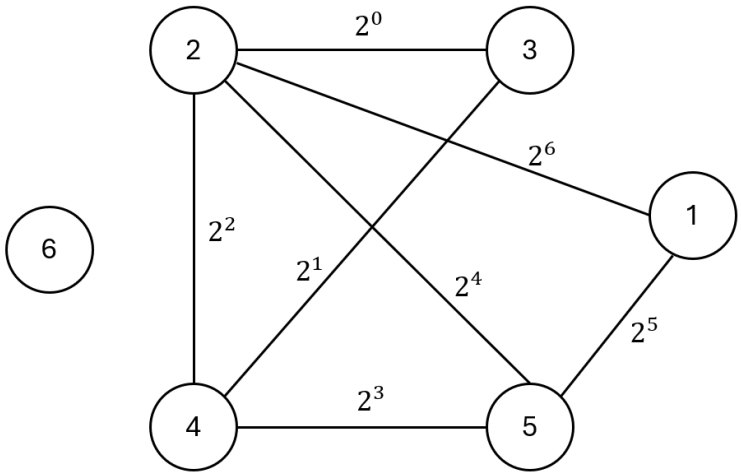

Figure 4: Illustration of the sample test case

The minimum travel time routes from location 1 are as follows:

- To location 2: $1 \to 5 \to 4 \to 3 \to 2$. The total travel time is $2^5 + 2^3 + 2^1 + 2^0 = 32 + 8 + 2 + 1 = 43$ minutes. Therefore, $d_2 = 43 \bmod 998244353 = 43$.
- To location 3: $1 \to 5 \to 4 \to 3$. The total travel time is $2^5 + 2^3 + 2^1 = 32 + 8 + 2 = 42$ minutes. Therefore, $d_3 = 42 \bmod 998244353 = 42$.
- To location 4: $1 \to 5 \to 4$. The total travel time is $2^5 + 2^3 = 32 + 8 = 40$. Therefore, $d_4 = 40 \bmod 998244353 = 40$.
- To location 5: $1 \to 5$. The total travel time is $2^5 = 32$. Therefore, $d_5 = 32 \bmod 998244353 = 32$.
- Location 6 is unreachable from location 1, so $d_6 = -1$.

Group 3

This group consists of Problems H, I, J and K.
These problems are challenging. Remember that difficulty is subjective.
Remember to read all problems and attempt the ones you feel most comfortable with. Good luck!

This page is intentionally left blank.

Problem H. Holy Division

Time limit: 3 s *Memory limit: 512 MB*

This is an interactive problem.

Make sure to flush the output **after each query**, or your program's verdict will be judged as Time Limit Exceeded. This works in various languages:

- Java: `System.out.println()` flushes automatically.
- Python: `print(..., flush = True)` flushes.
- C++: `cout << endl;` flushes automatically, in addition to writing a newline. If using `printf`, use `fflush(stdout)`.

Apollo, the god of prophecy, has hidden an integer n ($1 \leq n \leq 10^8$). Mortals are not allowed to know n directly, but Apollo may answer queries and perform divisions for you.

One of Apollo's prophecies states that an integer x is *sacred* if and only if x has **at most** 8 **positive divisors**.

Although mortals do not know Apollo's value of n, Apollo believes you are no mere mortal. Prove your worthiness by reducing n to a *sacred* value n' (where $n' \leq n$).

Interaction Protocol

For the purposes of this interaction, the jury acts on Apollo's behalf.

The interaction begins when you ask your first query. On your turn, you can make one of three types of queries:

- `gcd d` ($1 \leq d \leq 10^{18}$) followed by a newline. The jury then returns $\gcd(n, d)$. The value of n is unchanged by this query.
- `div d` ($1 \leq d \leq 10^{18}$) followed by a newline. If d divides n, the jury replaces n with n/d and returns `1`; otherwise n is unchanged and the jury returns `0`.
- `!` followed by a newline. This declares that the current value of n is *sacred*. **Your program should terminate immediately after this.**

You may ask a combined total of 7 queries of `gcd d` and `div d`.

In the event that you read `-1` instead of valid data, your program must terminate immediately. This indicates that your submission has made an invalid query or has otherwise received a Wrong Answer verdict.

Please turn over for the sample interaction.

The following sample interaction is meant to showcase the interaction between your program and the jury, but does not show how the answer was derived. Note that comments (marked by `#`) are there for readability and are not part of the actual interaction.

Sample Input	Sample Output

`0 # reply 1`
`1 # reply 2`
`1 # reply 3`
`16 # reply 4`
	`div 3 # query 1`
	`div 4 # query 2`
	`div 4 # query 3`
	`gcd 16 # query 4`
	`! # answer`

Note Apollo's integer was initially $n = 256$. The sample makes a total of 4 out of 7 queries.

- The first query attempts to divide n by 3. Since 3 is not a divisor of $n = 256$, n remains unchanged and the jury outputs `0`.
- The second query attempts to divide n by 4. Since 4 is a divisor of $n = 256$, $n \leftarrow 256/4 = 64$. After this query $n = 64$, and the jury outputs `1`.
- The third query attempts to divide n by 4. Since 4 is a divisor of $n = 64$, $n \leftarrow 64/4 = 16$. After this query $n = 16$, and the jury outputs `1`.
- The fourth query asks for $\gcd(n, 16)$. The jury outputs 16.

Finally, the interaction concludes with `!` because 16 has 5 positive divisors (the positive divisors of 16 are $[1, 2, 4, 8, 16]$), which is *sacred*.

Problem I. Infinite Polygon

Time limit: 2 s *Memory limit: 512 MB*

You are given an infinite sequence of points $p_0, p_1, p_2, \cdots$ defined as:

$$p_0 = (1, 0), \quad p_1 = (1, \frac{1}{2}), \quad p_i = (\frac{1}{i}, \frac{1}{i+1}) \text{ for } i \geq 1$$

The sequence continues indefinitely and approaches the point $(0, 0)$. See the figure on the next page for an illustration of the sequence and the polygon $\mathcal{P}$.

The polygon $\mathcal{P}$ is created by connecting the points in order with line segments

$$p_0 \to p_1 \to p_2 \to \cdots$$

The boundary is completed by adding the segment from $(0, 0)$ to p_0.

Answer q queries. Each query (n_x, d_x, n_y, d_y) represents a point

$$Q = (\frac{n_x}{d_x}, \frac{n_y}{d_y}).$$

For each query, compute the minimum Euclidean distance from Q to the region enclosed by $\mathcal{P}$. Thus, if Q lies inside or on the boundary of $\mathcal{P}$, the distance is 0.

Input

- The first line contains a single integer q ($1 \leq q \leq 2 \times 10^5$), the number of queries.
- q lines follow, each line containing four integers n_x, d_x, n_y, d_y ($-10^2 \leq n_x, n_y \leq 10^2$, $1 \leq d_x, d_y \leq 10^6$), describing the query point Q.

Output q lines. The i-th line should contain your answer for the i-th query. Your answer will be accepted if its absolute or relative error is at most 10^{-6}.

> **Note on Precision**: This problem involves floating-point computations. Although answers with absolute or relative error at most 10^{-6} are accepted, using higher-precision floating point types (such as `long double` in C++) is recommended.

Please turn over for the sample test cases.

Sample Input	Sample Output
13 9 10 1 4 1 2 -1 5 1 1 1 3 3 4 1 2 6 5 1 5 -1 10 0 1 -1 10 -1 10 1 100 1 100 38 1 1 1 -71 75 44 851407 -79 839169 35 925573 38 406091 44 579238 -72 71073 49 159480	0 0.2 0 0.079056941 0.2 0.1 0.141421356 0.0000707071 37.003378224 0.9466666681 0.0001014515 0 0.0010586112

Note Refer to Figure 5 for an illustration. This visualization includes the first four query points: $Q_1 = (\frac{9}{10}, \frac{1}{4})$, $Q_2 = (\frac{1}{2}, \frac{1}{5})$, $Q_3 = (1, \frac{1}{3})$ and $Q_4 = (\frac{3}{4}, \frac{1}{2})$.

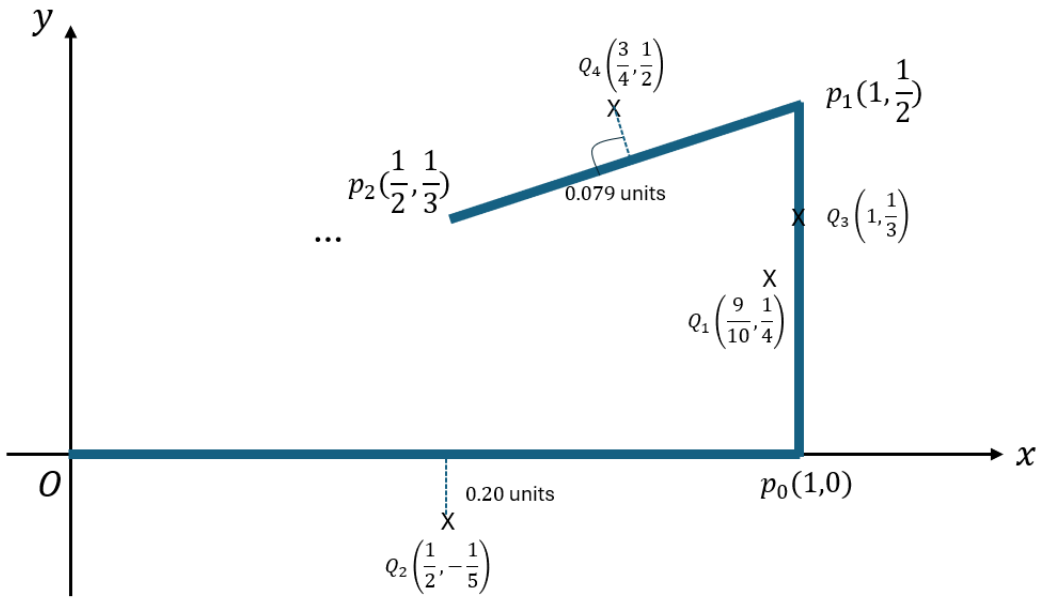

Figure 5: Illustration of polygon $\mathcal{P}$ and query points Q_1, Q_2, Q_3 and Q_4. O denotes the point $(0, 0)$. Thick lines denote the illustrated portion of the boundary of polygon $\mathcal{P}$. Query points Q_1, Q_2, Q_3, Q_4 are marked with an `X`. The minimum Euclidean distance to $\mathcal{P}$ for each point is illustrated by thin lines.

- Point Q_1 is inside $\mathcal{P}$, so output 0.
- Point Q_2 is nearest to the segment from $(0, 0)$ to p_0, with a distance of 0.20 units.
- Point Q_3 lies on segment p_0p_1, hence the answer is 0.
- Point Q_4 has a distance of approximately 0.0791 units from segment p_1p_2, which is the minimum distance to $\mathcal{P}$.

Problem J. Judicious Crime Response

Time limit: 2 s *Memory limit: 512 MB*

There are n streets in a city, numbered from 1 to n. For every $1 \leq i \leq n-1$, street i and $i+1$ are connected by a two-way road with travel time t_i units.

There is also one police station on each street. The i-th station is located on street i, has a forwarding time delay of f_i units, and a response time delay of r_i units.

Station s receives an emergency call reporting a crime at street d. The receiving station then chooses one of the following actions:

- **Forward the call**: If there exists a station k on a higher-indexed street ($k > s$) with shorter response time delay, that is, $r_k < r_s$, station s may forward the call to the **nearest** such station k^*. After a forwarding time delay of f_s time units, station k^* receives the forwarded call.
- **Respond to the call**: After a response time delay of r_s time units, a police car is dispatched from street s and travels by road to street d, incurring the corresponding road travel time.

The total response time is the elapsed time from the initial call until a police car arrives at street d. All stations choose actions to minimize this time.

The city receives q emergency calls of the form (s, d). For each call, find the minimum total response time if station s initially receives a call of a crime at street d.

Input

- The first line contains two integers n and q ($2 \leq n, q \leq 2 \times 10^5$), the number of streets n, and the number of queries q. There is one station on each street.
- The second line contains n integers $f_1, f_2, \cdots, f_n$ ($1 \leq f_i \leq 10^9$), the i-th integer denoting the forwarding time delay f_i of station i.
- The third line contains n integers $r_1, r_2, \cdots, r_n$ ($1 \leq r_i \leq 10^9$), the i-th integer denoting the response time delay r_i of station i.
- The fourth line contains $n-1$ integers $t_1, t_2, \cdots, t_{n-1}$ ($1 \leq t_i \leq 10^9$), the i-th integer denoting the travel time by road from street i to street $i+1$.
- The next q lines contain two integers s and d ($1 \leq s, d \leq n$), the i-th of these lines represents the i-th emergency call.

Output q lines. For each query (s, d), output the minimum total response time if station s first receives the call reporting a crime at street d.

Please turn over for the sample test case.

Sample Input	Sample Output
5 5 4 7 10000 2 6 9 6 8 10000 1 2 3 10000 6 1 5 1 3 3 4 4 5 5 1	12 13 10007 3 10012

Note There are 5 streets and police stations, and 5 emergency calls.

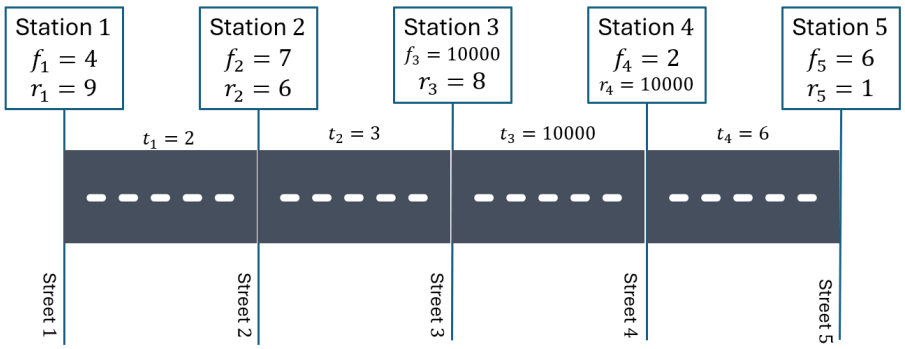

Figure 6: Illustration of Sample Test Case

For the first emergency call $(1, 5)$, the call is initially received by station 1 and the crime is at street 5. There are 3 possible ways to respond to the call.

One possible sequence of events is as follows:

1. Station 1 forwards the call to station 2, incurring a forwarding delay of 4 units.
2. Station 2 forwards the call to station 5, incurring a forwarding delay of 7 units.
3. Station 5 incurs a response time delay of 1 unit.

The total response time of this sequence of events is $4 + 7 + 1 = 12$ time units.

Another possible sequence of events is as follows:

1. Station 1 forwards the call to station 2, incurring a forwarding time delay of 4 units.
2. Station 2 incurs a response time delay of 6 units and dispatches a police car to street 5. The car takes $3 + 10000 + 6 = 10009$ time units to reach street 5.

The total response time of this sequence of events is $4 + 6 + 10009 = 10019$ time units.

Lastly, the only other sequence of events is as follows:

1. Station 1 incurs a response time delay of 9 units and dispatches a police car to street 5. The car takes 10011 units to reach street 5.

The total response time of this sequence of events is $9 + 10011 = 10020$ time units.

Therefore, the minimum total response time for the first call is 12 units.

Problem K. Killed by Sorting

Time limit: 2 s *Memory limit: 512 MB*

A suspicious streamer named *TLE Creators* proposed an algorithm to sort a permutation. Note that the code below uses one-indexed positions.

```
Function CyclicSort(arr):
     pos = 1
     WHILE (arr is not sorted) DO
        index = find the index of the minimum element in arr[pos...n]
        shifts = index - pos
        FOR (i = 1...shifts) DO
            cyclically shift left arr[pos...n] by one position
        END
        pos = pos + 1
     END
END
```

Let's consider an example by performing `CyclicSort` on the permutation $p = [2, 4, 1, 3, 5]$.

- `pos` $= 1$. The minimum element is at position 3 ($p_3 = 1$), and the subarray $p_{1\cdots5}$ is cyclically shifted to the left by one position twice, incurring 2 operations.
  The resulting array is $p^{(1)} = [1, 3, 5, 2, 4]$.
- `pos` $= 2$. The minimum element is at position 4 ($p^{(1)}_4 = 2$), and the subarray $p^{(1)}_{2\cdots5}$ is cyclically shifted to the left by one position twice, incurring 2 operations.
  The resulting array is $p^{(2)} = [1, 2, 4, 3, 5]$.
- `pos` $= 3$. The minimum element is at position 4 ($p^{(2)}_4 = 3$), and the subarray $p^{(2)}_{3\cdots5}$ is cyclically shifted to the left by one position once, incurring 1 operation.
  The resulting array is $p^{(3)} = [1, 2, 3, 5, 4]$.
- `pos` $= 4$. The minimum element is at position 5 ($p^{(3)}_5 = 4$), and the subarray $p^{(3)}_{4\cdots5}$ is cyclically shifted to the left by one position once, incurring 1 operation.
  The resulting array is $p^{(4)} = [1, 2, 3, 4, 5]$.
- Since the array is sorted, the algorithm terminates.

TLE Creators proved that `CyclicSort` always sorts any permutation in ascending order, but they are having difficulty determining the number of cyclic left-shift operations that `CyclicSort` will perform. They are going live right after your contest, and desperately need your help!

Given a permutation p of length n, please help *TLE Creators* determine the number of cyclic left-shift operations that `CyclicSort` will perform.

Please turn over for the input and output specifications.

Input The first line contains a single integer t ($1 \le t \le 10^3$), the number of test cases. For each test case,

- The first line contains a single integer n ($1 \le n \le 2 \times 10^5$), the length of the permutation p.
- The second line contains n integers $p_1\ p_2\ \cdots\ p_n$ ($1 \le p_i \le n$), where p_i is the i-th value of the permutation p. It is guaranteed that p is a permutation of $1, 2, \cdots, n$.

The sum of n over all test cases will not exceed 2×10^5.

Output t lines. For each test case, output a single integer, the number of cyclic left-shift operations performed by `CyclicSort`.

Sample Input	Sample Output
4 5 2 4 1 3 5 6 2 3 1 4 5 6 6 6 5 4 3 2 1 7 1 2 3 4 5 6 7	6 5 15 0

Note There are 4 test cases.

- The first test case is illustrated in the statement. The total number of shift operations is $2 + 2 + 1 + 1 = 6$
- In the second test case, the array transitions and cyclic left shift counts are as follows:
  - $p^{(1)} = [1, 4, 5, 6, 2, 3]$, cyclic left shift count $= 2$
  - $p^{(2)} = [1, 2, 3, 4, 5, 6]$, cyclic left shift count $= 3$

  Therefore the total number of shifts is 5.