The 2025 ICPC Asia Bangkok Regional Contest

Problem Set

Araya Luanseng
Atittarn Buathep
Jonathan Irvin Gunawan
Mattanyu Tangngekkee
Natapong Sriwatanasakdi
Papangkorn Apinyanon
Poonyapat Sriroth
Supakorn Kijwattanachai

Department of Computer Engineering, Faculty of Engineering
Department of Statistics, Faculty of Commerce and Accountancy
Department of Mathematics and Computer Science, Faculty of Science
Chulalongkorn University

This page is intentionally left blank.

A Among Us

1s, 256MB **(Interactive Problem)**

Aboard the spaceship *Skeld* are you — the commander — along with N crewmates. The voyage from Earth to Polus was meant to be peaceful, until it wasn't. A series of strange incidents occurs: oxygen runs low, the reactor nearly melts down, the lights flicker, and the doors jam shut. Suspicion spreads — there might be impostors among us.

Moments of crisis were handled with the crewmates' unity, but the tension lingered. Each crewmate now secretly suspects exactly **one** crewmate of being the impostor. Remarkably, **no crewmate is suspected by more than one person**. Formally, there exists a permutation P of length N, where $P[i]$ is the crewmate whom the ***i***-th crewmate suspects.

Overseeing this chaos from the commander's room, you, however, are not actually concerned with finding the impostors. Rather**, your only goal is to determine the hidden permutation P.**

Naturally, to achieve this, you may call **emergency meetings**. In each meeting, you can choose the order in which the crewmates speak.

Given a crewmate speaking, they will accuse the crewmate they suspect. If the speaker has not yet been marked as ***sus***, their accusation is valid, and the accused crewmate becomes ***sus***. However, if the speaker is already ***sus***, their accusation is ignored.

Unfortunately, as the commander, you cannot attend the meetings yourself. Instead, after each meeting, the ship's log reports the ***sus*** status of all crewmates as a binary array, where 1 indicates ***sus*** and 0 indicates ***not sus***.

Since time is limited, you may hold at most 400 emergency meetings. Your task is to determine the hidden permutation P within this limit.

INTERACTION

Your program first reads a single integer N $(2 \le N \le 100)$ — the number of crewmates.

Then, you may repeatedly perform queries (call emergency meetings). To make a query, print a line in the following format:

- $?\ q_1\ q_2\ \dots\ q_N$

Here, $q_1\ q_2\ \dots\ q_N$ is the order in which the crewmates will speak during the meeting. The sequence must be a permutation of length N.

After printing a query, flush the output and read a line containing N integers: $s_1\ s_2\ \dots\ s_N$

Here, s_i indicates the status of the i-th crewmate after the meeting ends:

- $s_i = 0$ — the i-th crewmate is not ***sus***.
- $s_i = 1$ — the i-th crewmate is ***sus***.

Each emergency meeting is independent; all crewmates start as not sus.

When you have determined the hidden permutation P, print it in the following format:

- ! p_1 p_2 … p_N

Printing the answer does **not** count as a query. After printing the final answer, your program should immediately terminate. Failure to do so may result in an undefined verdict.

You may perform at most **400 queries**. If your program performs more than 400 queries or outputs an invalid format, you may receive a Wrong Answer verdict.

The interactor is **not adaptive**, which means that the hidden permutation does not depend on the queries you make.

After printing a query or an answer, do not forget to output the end of line and flush the output. Otherwise, you may get an `Idleness Limit Exceeded` verdict. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python;

EXAMPLE

INPUT	**OUTPUT**
`6`	
	`? 1 2 3 4 5 6`
`1 0 1 1 1 0`	
	`? 2 4 3 1 5 6`
`0 0 1 1 1 1`	
	`? 6 5 4 3 2 1`
`1 1 1 0 0 1`	
	`? 3 5 4 2 1 6`
`0 1 1 1 0 1`	
	`! 4 5 3 2 6 1`

NOTES

For the first emergency meeting, the permutation P is [4,5,3,2,6,1] and the accusations go as follows:

- Crewmate 1 accuses crewmate 4. Crewmate 4 becomes *sus*.
- Crewmate 2 accuses crewmate 5. Crewmate 5 becomes *sus*.
- Crewmate 3 accuses crewmate 3. Crewmate 3 becomes *sus*.
- Crewmate 4 accuses crewmate 2, but they are already *sus*, so it is ignored.
- Crewmate 5 accuses crewmate 6, but they are already *sus*, so it is ignored.
- Crewmate 6 accuses crewmate 1. Crewmate 1 becomes *sus*.
- The ship reports the status of the crewmates as 1 0 1 1 1 0.

B Bring It To Back

1s, 256MB

Jack is playing a game with his cat, **Salmon**. On a shelf, Jack has comic books labeled from volume 1 to volume N, arranged in some order from left to right. All books are identical in size.

Jack and Salmon will repeat the following sequence of moves **exactly** M times (where M may be zero):

1. Jack chooses one comic book that is **not the rightmost book** and knocks it to the ground.
2. He then closes the gap by shifting all books to the right of that position leftward to fill the space, leaving the empty spot at the rightmost end of the shelf.
3. Finally, Salmon picks up the fallen book and places it into the gap at the rightmost position.

Your task is to determine the **lexicographically largest initial arrangement** of the books such that, after performing exactly M sets of moves as described, there exists a sequence of choices for Jack that results in the books being sorted in ascending order from volume 1 to volume N from left to right.

Two different arrangements A and B of N books are compared lexicographically as follows: A is said to be **lexicographically larger** than B if, at the first position i from left side where they differ, $A_i > B_i$.

INPUT

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

Each test case consists of a single line containing two integers N and M $(2 \le N \le 10^5, 0 \le M \le 10^9)$ — the number of comic books on the shelf and the number of sets of moves performed, respectively.

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

OUTPUT

For each test case, print N integers — the **lexicographically largest** initial order of the books that will result in the volumes being arranged as $1,2,\ldots,N$ after exactly M moves if Jack chooses the books optimally.

EXAMPLE

INPUT	OUTPUT

`2`
`3 1`
`5 2`
	`3 1 2`
	`5 4 1 2 3`

NOTE

In the first test case, Jack can knock down book number 3, and Salmon inserts it at the rightmost position, resulting in 1 2 3.

Another possible initial arrangement is 1 3 2, but this arrangement is not lexicographically largest.

The initial arrangement 1 2 3 is not possible: knocking down book 1 gives 2 3 1, and knocking down book 2 gives 1 3 2.

C Challenge to the Reader

1s, 256MB

A mathematician from ancient times left behind a riddle for future generations. Carved into stone lies a challenge about numbers and operations: Any number can be represented using a sequence of $+$ and $-$ signs applied to the numbers 1, 2, 3, ...

You, a bright and curious mind from the future, decide to take up this challenge.

Given an integer X, your task is to express it in the form $1 \pm 2 \pm 3 \pm 4 \pm \cdots \pm N$ so that the result equals X and $\boldsymbol{N}$ **is the smallest possible number**.

INPUT

The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases.

Each testcase contains a single line of one integer X $(|X| \le 2 \cdot 10^5)$.

It is guaranteed that the sum of $|X|$ over all test cases is at most $2 \cdot 10^5$.

OUTPUT

For each testcase:

First, print a line with an integer N

Then print one valid expression in the format $1 \pm 2 \pm 3 \pm 4 \pm \cdots \pm N$. No spaces are allowed, and the sequence must include all numbers from 1 to N.

If there are many expressions, output any of them.

EXAMPLE

INPUT	OUTPUT

```
5
0
-2
1
4
-6
```
	```
	3
	1+2-3
	4
	1-2+3-4
	1
	1
	4
	1+2-3+4
	7
	1+2+3-4+5-6-7
	```

This page is intentionally left blank.

D Dungeons and Dragons

1s, 512MB

Audrey and Bastian are about to start on an epic dungeon crawl where they'll face R monsters ($1 \le R \le 10^9$). Each monster has a health point (HP) value between 0 and N ($1 \le N \le 5 \times 10^5$).

Game Rules

The battle proceeds in turns, with Audrey going first:

- On each turn, the current player selects a monster with HP > 0
- If the chosen monster has HP $= x$, the player can choose to deal damage d where $1 \le d \le p[x]$. It's guaranteed that $1 \le p[x] \le x$ for all x.
- The monster's HP becomes $x - d$.
- The player who defeats the last monster (reduces the last non-zero HP to 0) wins

Both players play optimally.

The Setup Phase

Audrey's Preparation: Audrey, being a cunning adventurer, prepares the dungeon in advance. She sets the initial HP of all R monsters to values $a_1, a_2, \ldots, a_R$ where $0 \le a_i \le N$. Audrey configures the monsters such that she is guaranteed to win if the game proceeds with these HP values.

Bastian's Sabotage: Bastian discovers Audrey's scheme and decides to sabotage exactly one monster before the battle begins. He chooses one monster with HP $= a_i$ where $a_i \ge K$ and decreases its HP by K, making its new HP equal to $a_i - K$.

Bastian's goal is to turn the tables and guarantee his own victory. If there's no way for Bastian to choose a monster that guarantees his win, Bastian abandons the dungeon (the game doesn't happen).

You have to determine how many distinct configurations $(a_1, a_2, \ldots, a_R)$ are possible on the final day (after Bastian's sabotage, if the game proceeds).

Important: Count configurations based on the HP values after Bastian's modification, not the initial values Audrey set.

Two configurations are considered different if they contain different HP values or the same HP values in different orders (e.g., $(1,3,2) \neq (1,2,3)$).

INPUT

The first line contains three integers N, K, and R — the maximum HP value, the reduction amount, and the number of monsters.

The second line contains N integers $p[1], p[2], \ldots, p[N]$ — where $p[i]$ is the maximum damage that can be dealt to a monster with i HP.

OUTPUT

Output a single integer — the number of possible monster HP configurations on the final day, modulo 998244353.

EXAMPLE

INPUT	OUTPUT

`10 3 1`
`1 2 1 2 1 3 4 5 4 5`
	`2`

`10 3 3`
`1 2 1 2 1 3 4 5 4 5`
	`194`

NOTE

In the first test case, with $R = 1$ monster, the possible configurations on the final day are single-monster configurations. The valid final configurations are (3) and (5), giving us 2 possible configurations.

For the second test case, one example of a valid final day configuration is (1,7,2). This configuration could arise from:

- Audrey initially chose (1,7,5), then Bastian reduced the monster with HP = 5 by K=3 to get HP = 2
- Audrey initially chose (1,10,2), then Bastian reduced the monster with HP = 10 by K=3 to get HP = 7

In both cases, the initial configuration guarantees Audrey's win, but after Bastian's sabotage, the final configuration (1,7,2) guarantees Bastian's win. The total number of such valid final configurations is 194.

E Elena and Travel Pass

2s, 256MB

Elena the Ashen Witch has decided to spend some time in a magical country and is now searching for a place to stay. The country consists of N cities, numbered from 1 to N, connected by M **one-way streets**. Since flying magic has been banned for intercity travel, Elena must rely on these streets to move from one city to another.

Each street is described by four integers u, v, p, and h — meaning there is a road leading from city u to city v that requires a pass level of p and takes h hours to traverse. To use a street, Elena needs a travel pass. If she holds a pass of level p, she can use any streets that require a pass level less than or equal to p. There may be multiple streets from city u to city v.

Elena has prepared Q questions for her journey, each falling into one of the following two types:

- **Type 1**: Elena is considering staying in city u and wants to know the **minimum pass level** required so that, with that pass, the travel time from city u to any city does not exceed h hours.
- **Type 2**: Elena is choosing where to stay and wants to find a city such that, from that city, she can reach any city within h hours while requiring the **lowest possible pass level**. If multiple cities achieve the same minimum requirement, she will choose the one with the **smallest city number**, since the average cost of living there is lower.

INPUT

The first line contains three integers N, M, and Q $(2 \leq N \leq 100, 1 \leq M \leq 10^4, 1 \leq Q \leq 10^5)$ — the number of cities, the number of streets, and the number of queries, respectively.

Each of the following M lines contains four integers u, v, p, and h $(1 \leq u, v \leq N, u \neq v, 1 \leq p \leq 10^9, 1 \leq h \leq 10^9)$, indicating that there is a road from city u to city v that requires a pass level of p and takes h hours to travel.

The next Q lines describe the queries, each following one of the two formats below:

- Type 1: `1 u h` $(1 \leq u \leq N, 1 \leq h \leq 10^{12}$, `u` and `h` are integers).
- Type 2: `2 h` $(1 \leq h \leq 10^{12}$, `h` is integer).

OUTPUT

Output For each query i $(1 \leq i \leq Q)$, output the answer to the i-th query.

If the query is of **Type 1**, print a single integer — the minimum pass level required. If there is no valid answer, print -1.

If the query is of **Type 2**, print two integers — the city and the minimum pass level required. If it is impossible, print `-1 -1`. If there is more than one city achieving the minimum pass level, print the one with the **lowest** number.

EXAMPLE

INPUT	OUTPUT

```
5 6 4
1 2 1 3
2 3 1 1
3 4 1 2
4 5 1 3
5 1 1 2
1 3 2 1
1 1 9
2 8
1 1 8
1 1 5
```
	```
	1
	2 1
	2
	-1
	```

NOTE

In the first question, Elena decides to stay in city 1. With the pass level 1, city 5 takes the most time out of other cities with 9 hours — still in time.

In the second question, with a pass level 1, Elena can travel from city 2 to any city in 8 hours. City 5 also achieves the same time, but the number is higher.

In the third question, with a pass level 2, Elena can travel from city 1 to city 2 in 3 hours, to city 3 in 1 hour, city 4 in 3 hours, and city 5 in 6 hours.

In the fourth question, Elena cannot travel from city 1 to city 5 in less than 6 hours with any pass level.

F

Festival Stroll

1s, 256MB

With the quarantine measures lifted, Jack's hometown decides to hold a festival. Jack is still cautious about meeting too many people, so he wants to plan his route through the festival to maximize his happiness using a simple strategy.

The festival has N stalls, numbered from 1 to N. If Jack enters the i-th stall, he meets p_i people and gains happiness h_i. He walks from stall 1 to stall N in order and does not revisit any stall. At each stall, he can either enter it or walk past it.

It is guaranteed that $p_i \geq p_{i+1}$ for all $i < N$, since earlier stalls tend to draw larger crowds.

Jack wants to meet at most P people in total. He also considers only stalls whose happiness exceeds a certain threshold value threshold. The threshold should be a non-negative integer.

Formally, let `ht` be Jack's total happiness and `m` be the number of people he has met. Jack will follow the following strategy.

Jack's strategy (pseudo-code):

```
ht = 0 // initially, the total happiness is zero
m = 0 // initially, he meets zero person
for i = 1 to N:
    if h[i] > threshold and m + p[i] <= P:
        enter stall i
        ht = ht + h[i]
        m = m + p[i]
    else:
        skip stall i
```

Jack wants to choose the value of *threshold* that maximizes his total happiness `ht`. If multiple thresholds achieve this maximum, he wants the **minimum** such threshold.

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:

The first line contains two integers N and P ($1 \leq N \leq 2 \cdot 10^5, 1 \leq P \leq 10^{18}$) — the number of stalls and the maximum number of people Jack wants to meet.

Each of the next N lines contains two integers h_i and p_i ($1 \leq h_i \leq 10^9, 1 \leq p_i \leq 10^{18}$) — the happiness gained and the number of people met at the i-th stall.

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

Additionally, for each test case, the sum of all p_i does not exceed 10^{18} and $p_j \geq p_{j+1}$ for all $j < N$.

OUTPUT

For each test case, print two integers — the maximum total happiness Jack can achieve and the minimum threshold to achieve the total happiness.

EXAMPLE

INPUT	OUTPUT

```
2
4 15
4 6
1 5
3 3
3 1
4 14
4 6
1 5
3 3
3 1
```
	```
	11 0
	10 1
	```

NOTE

In the first test case, Jack can enter every stall without exceeding the maximum number of people he wants to meet. Therefore, the optimal threshold is 0.

In the second test case, if the threshold is set to 0, Jack will skip stall 3, resulting in a total happiness of 8. If the threshold is increased to 1, he will instead skip stall 2 due to the threshold condition, resulting in a total happiness of 10, which is the maximum possible. Threshold value 2 also resulted in total happiness of 10, but it is not minimal.

G Galactic Adventure Agency

3s, 256MB

In a galaxy far, far away, there are N planets. The i-th planet is located at coordinates (x_i, y_i, z_i). The planets are connected by a galactic railway system consisting of $N - 1$ **bi-directional** railways, forming a tree structure.

As an agent of the *Galactic Adventure Agency*, you are planning a promotional campaign. For this campaign, you must choose **two planets** to highlight. Wanderers can journey between these two planets in two different ways:

- By **galactic train**: Traveling along the unique simple path that connects the two planets in the railway system. Each railway has an associated **satisfaction score** (*it can be negative*), and the total satisfaction is the sum of the scores along the path.
- By **private rocket**: Traveling directly between the two planets along the x, y, and z axes. The satisfaction score for this option is equal to the **Manhattan distance** between the two planets, i.e. $|x_i - x_j| + |y_i - y_j| + |z_i - z_j|$.

To please both types of tourists, you want to select a pair of planets u, v that maximizes the combined satisfaction value, defined as the **sum** of the train satisfaction and the rocket satisfaction between them.

Your task is to find this maximum combined satisfaction value. If there is no pair of planets that has positive combined satisfaction value, the answer must be 0.

INPUT

The first line of input contains one integer N $(2 \le N \le 2 \cdot 10^5)$ — the number of planets.

Each of the next $\boldsymbol{N} - \mathbf{1}$ lines contains three integers, u, v, and w, — a railway between planet u and planet v with a satisfaction score of w $(1 \le u, v \le N;\ u \ne v;\ |w| \le 10^9)$. It is guaranteed that these edges form a tree.

Each of the next $\boldsymbol{N}$ lines contains three integers, x, y, and z — the coordinates of the planet u $(1 \le x, y, z \le 10^{14})$.

OUTPUT

A line contains one integer --- the maximum combined satisfaction value.

EXAMPLE

INPUT	OUTPUT

`5`
`1 2 -2`
`2 4 5`
`3 4 1`
`4 5 -5`
`8 3 2`
`7 6 1`
`8 6 2`
`3 6 3`
`1 1 1`
	`12`

NOTE

For the example, if you choose planet 1 and planet 4, the train satisfaction is equal to $-2 + 5 = 3$ and the rocket satisfaction is $|8 - 3| + |3 - 6| + |2 - 3|$, the combined satisfaction is equal to $9 + 3 = 12$, which can be shown to be the maximum.

H Home Workout Playlist

4s, 256MB

Life feels dull when you cannot go outside or meet your favorite people. Fortunately, your favorite bands keep releasing new hits this year, such as *Ussewa*, *Wan Koei Tuen*, and *Thought Crime*. Therefore, you decide to work out at home with your playlist!

You have a playlist of N songs, numbered from 1 to N, representing the order in which they are played. Each song i has a **hypeness value** A_i.

While exercising, you only want to listen to songs that match a certain pattern of rising excitement. You may choose to **skip some songs**, keeping the remaining ones in their original order.

Formally, let $S = [S_1, S_2, \ldots, S_k]$ be a subsequence of $[1, 2, \ldots, N]$ representing the indices of unskipped songs. Your task is to find the **longest** possible subsequence S such that:

- The hypeness strictly increases: $A_{S_i} > A_{S_{i-1}}$ for all $i \geq 2$.
- The increase in hypeness also strictly increases: $A_{S_i} - A_{S_{i-1}} > A_{S_{i-1}} - A_{S_{i-2}}$ for all $i \geq 3$.

In other words, both the hypeness values and the gaps between them must form strictly increasing sequences. Find the maximum possible length of such a subsequence S.

INPUT

The first line contains a single integer N ($1 \leq N \leq 5 \cdot 10^4$) — the number of songs in the playlist.

The second line contains N integers $A_1, A_2, \ldots, A_N$ ($1 \leq A_i \leq 10^5$) — the hypeness values of the songs.

OUTPUT

On the first line, print one integer k — the number of unskipped songs.

On the second line, print k integers — the indices of the unskipped songs in ascending order.

If there are multiple valid lists of unskipped songs, you may print any of them.

EXAMPLE

INPUT	OUTPUT
5 2 1 3 4 6	3 3 4 5

NOTE

Other valid lists of unskipped songs include [1,3,5], which correspond to hypeness values [2,3,6], and [2,3,5], which correspond to hypeness values [1,3,6]. You may output any of them.

I ICPC Extractor

1s, 256MB

Myth generation has begun !

Ina is given a string S of length N, consisting only of the characters 'I', 'C', and 'P'.

Her task is to repeatedly extract the substring "ICPC" from S as many times as possible. Each time she extracts "ICPC", she must choose four characters from S in order (not necessarily contiguous, but respecting the original order) that form "ICPC". After each extraction, the chosen characters are removed from the string. The remaining characters close the gap and form the new string for the next extraction.

You must help her determine the maximum number of times she can extract "ICPC" and the indices (1-indexed) of the characters removed from the original string S for each extraction.

INPUT

The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases.

Each testcase contains one single line of the string S, consisting only of the characters 'I', 'C', and 'P'.

It is guaranteed that the sum of the length of S over all test cases is at most $2 \cdot 10^5$.

OUTPUT

For each testcase:

First, print a line with an integer K — the maximum number of extractions of "ICPC".

Then print K lines, each containing four integers — the positions of the characters removed during that extraction, based on the original string.

If there are many ways to extract, output any of them.

EXAMPLE

INPUT	OUTPUT

```
4
PICPPC
ICPICPCCI
CIPCICICPPCCP
IPCC
```
	```
	1
	2 3 5 6
	2
	1 2 3 7
	4 5 6 8
	2
	2 4 9 11
	5 6 10 12
	0
	```

This page is intentionally left blank.

J Joyeuse

5s, 256MB

At a grand and dazzling dance party, there are N guests, each with their own dance skill power a_i.

When two guests pair up on the dance floor, their performance score is equal to the **square root of the sum of their skill powers**. For example, if one guest has a skill power of 4 and another has 5, their duet would have a performance score of $\sqrt{4+5} = 3$.

As the host of this vivid celebration, you wish to measure the collective joy — the total performance score of every possible pair of guests dancing together exactly once.

Your task is to determine this total score.

INPUT

The first line contains a single integer n (2≤n≤200000) — the number of guests.

The second line contains N integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$ — the dancing skill powers of the guests.

OUTPUT

A single real number — the total performance score at the party.

The answer is considered correct if the relative error does not exceed 10^{-6}.

EXAMPLE

INPUT	OUTPUT

`5`
`7 3 11 2 17`
	`39.009712`

NOTE

All possible pairs and their performance scores are:

Guest 1 and guest 2: $\sqrt{7+3}$ ≈3.162277660

Guest 1 and guest 3: $\sqrt{7+11}$ ≈4.242640687

Guest 1 and guest 4: $\sqrt{7+2}$ =3

Guest 1 and guest 5: $\sqrt{7+17}$ ≈4.898979486

Guest 2 and guest 3: $\sqrt{3+11}$ ≈3.741657387

Guest 2 and guest 4: $\sqrt{3+2}$ ≈2.236067977

Guest 2 and guest 5: $\sqrt{3+17}$ ≈4.472135955

Guest 3 and guest 4: $\sqrt{11+2}$ ≈3.605551275

Guest 3 and guest 5: $\sqrt{11+17}$ ≈5.291502622

Guest 4 and guest 5: $\sqrt{2+17}$ ≈4.358898944

Adding them all up gives approximately 39.009712.

K Kickshot Tournament

1s, 256MB

Welcome to the Kickshot Tournament, part of the prestigious *International Cue & Pocket Championship (ICPC)*! This year is a special year because they decide to host the tournament in the virtual world instead of real life!

You are playing on a rectangular pool table of size $(R - 1) \times (C - 1)$. On every lattice point of the table lies a virtual coin, forming a total of $R \times C$ coins. The coordinate system is 1-indexed, with the coin at row 1, column 1 positioned at the **bottom-left** corner. **Rows** increase **upward**, and **columns** increase **to the right**.

A ball is placed on the coin located at **row** M, **column** N, and is shot at a perfect 45° angle in the **up-right** direction. Initially, for each millisecond, the ball moves one unit upward (toward larger row numbers) and one unit rightward (toward larger column numbers). When the ball hits the top or bottom wall, it flips its vertical direction; when it hits the left or right wall, it flips its horizontal direction. There is no energy loss — the ball continues bouncing perfectly until it lands in one of the four corner pockets.

Each time the ball passes through a coin, that coin immediately disappears from the table. Collecting a coin does not affect the trajectory of the ball. The coin at row **M** column **N** is immediately collected when shot. The process continues until the ball reaches a corner, where it finally stops.

Your task is to determine how many coins the ball collects in total, including both the starting coin and the final coin at the pocket.

INPUT

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

For each test case, there is one line that contains four integers M, N, R, and C $(3 \le R, C \le 10^9, 1 \le M < R, 1 \le N < C, (M, N) \ne (1,1))$ indicating the starting position of the ball and the size of the table.

OUTPUT

For each test case, output one line that contains a single integer X — the number of coins the ball collects.

If the ball never reaches any of the corners, output -1.

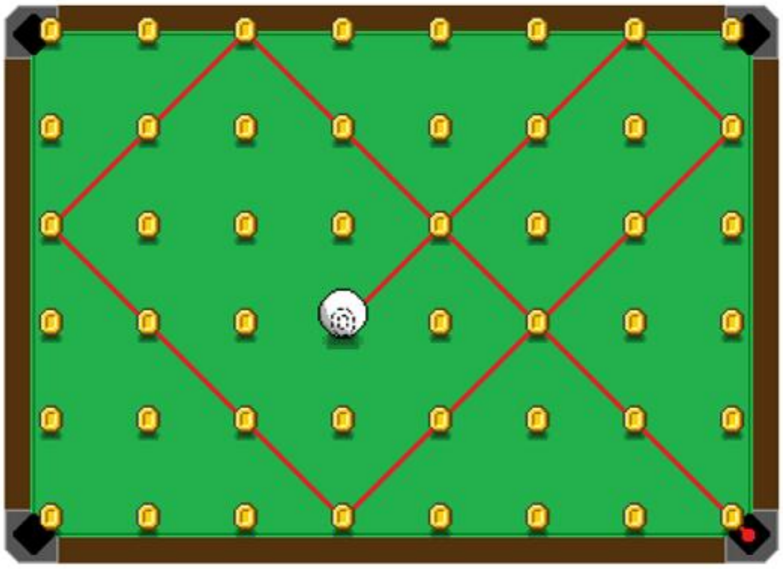
EXAMPLE

INPUT	OUTPUT

3
3 4 6 8
2 3 6 7
4 3 7 10
	17
	5
	-1

NOTES

The first testcase can be visualized as follows:



L Laser

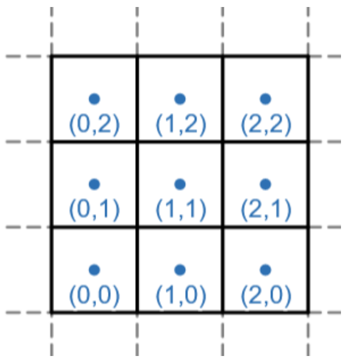
2s, 1024MB

A mega-scale project to convert laser light into usable energy for general purposes has begun in the almost infinite experiment area with laser-amplifying glass panes.

The experiment was conducted by placing a laser emitter in one room. After that, the N laser receivers were placed in room coordinates (x_i, y_i); each room has a glass pane on the vertical and horizontal side which is shared between adjacent rooms.

- When the laser passes through each vertical glass pane, the laser intensity is increased by A units.
- When the laser passes through each horizontal glass pane, the laser intensity is increased by B units.

We can choose to activate only 1 laser receiver which further amplifies the laser intensity by c_i units when the laser stops at that receiver, generating energy in proportion to the final laser intensity. To prevent accidental damage from the experiment, all equipment must be placed in the center of the room, and the laser must start with a horizontal or vertical trajectory, and the laser path must end on the activated laser receiver.



Furthermore, the administrator of the project has provided 1 L-shaped reflector to change the trajectory of the laser from horizontal to vertical or vice versa. You want to run Q simulation tests to provide the information on the minimum possible laser intensity given the starting point for the laser at coordinates (s_i, t_i); your task is to come up with the resulting laser intensity.

INPUT

The first line contains four integers N, Q, A, B $(0 \leq A, B \leq 10^9; 1 \leq N, Q \leq 100{,}000)$ — number of laser receivers, simulations, amplification power of vertical and horizontal glass panes.

Next, N lines contain x_i, y_i, c_i $(0 \leq x_i, y_i \leq 10^9; 0 \leq c_i \leq 10^{18})$ — laser receiver coordinates in the Cartesian plane and laser amplification power.

Next, Q lines contain s_i, t_i $(0 \leq s_i, t_i \leq 10^9)$ — laser emitter starting coordinates in the Cartesian plane in each simulation.

OUTPUT

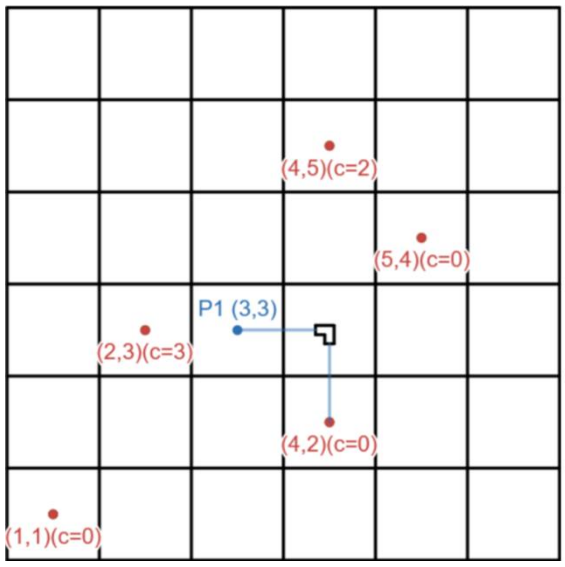
Print Q lines, each line contains a single integer — the minimum possible laser intensity for the simulation.

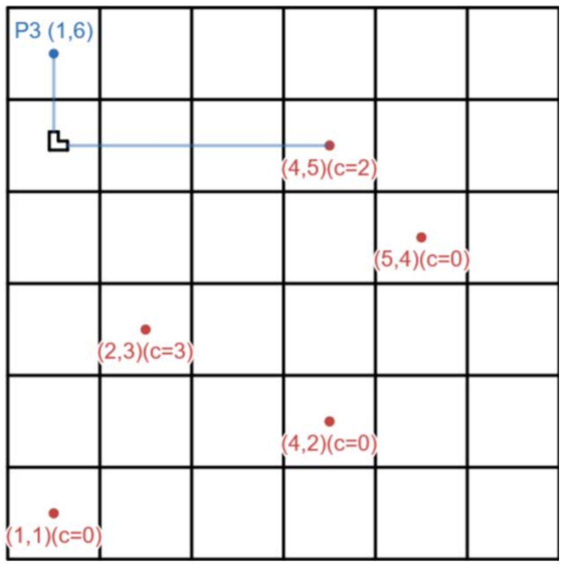
EXAMPLE

INPUT	OUTPUT

5 3 1 2
1 1 0
4 2 0
2 3 3
5 4 0
4 5 2
3 3
4 1
1 6
	3
	2
	7

Note





Let the red point and blue point be the laser receiver and the laser emitter in each simulation respectively. The above figure shows the arrangement for minimum possible laser intensity for the first and third simulations.

For the first simulation, the L-shaped reflector can be placed on coordinate (3,4); by activating the second receiver, the minimum final laser intensity is $1 + 2 + 0 \ = \ 3$.

For the second simulation, the L-shaped reflector is not used; by activating the second receiver, the minimum final laser intensity is $2 + 0 \ = \ 2$.

For the third simulation, the L-shaped reflector can be placed on coordinate (1,5); by activating the fifth receiver, the minimum final laser intensity is $2 + 1 + 1 + 1 + 2 \ = \ 7$.

M Merticulous Manipulation

1s, 256MB

The Marvelous Maximillian is a stage magician famous for his grand finale, *The Perfect Permutation*. He claims he can arrange N cards, numbered 1 to N, into any permutation P requested by the audience.

As his apprentice, you know his secret: it's a precise N-step procedure. The trick starts with an empty pile. For each card i from 1 to N, in order:

1. Maximillian places card i on top of the pile.
2. He immediately asks you for a secret number, x_i (where $1 \le x_i \le i$). He takes the top x_i cards from the pile and moves them, in order, to the bottom. For example, if the pile is [3, 1, 2] (top to bottom) and $x = 2$, the pile becomes [2, 3, 1].

Tonight, an audience has demanded that the final pile be in the specific order P, where P_1 is the top card and P_N is the bottom.

Maximillian is on stage and looking at you. You must provide the correct sequence of cut numbers, $x_1, x_2, \dots, x_N$ to produce the permutation P and save the show.

INPUT

The first line contains a single integer N $(2 \le N \le 200000)$ — the number of cards.

The second line contains N integers $P_1, P_2, \dots, P_N$ $(1 \le P_i \le N)$ — the permutation demanded by the audience.

OUTPUT

Print one line containing N integers — the sequence of cut numbers, $x_1, x_2, \dots, x_N$.

EXAMPLE

INPUT	OUTPUT
5 1 2 3 4 5	1 1 1 1 1
4 4 1 3 2	1 2 2 4

Note

For the first example, every cut is just taking the top card and putting it in the bottom.

For the second example, the pile after each cut is as follows.

- After first cut: [1]

- After second cut: [2, 1]
- After third cut: [1, 3, 2]
- After fourth cut: [4, 1, 3, 2]

N No Distance is Too Far Apart

1s, 256MB

Alice and Bob are standing somewhere in a vaccination queue of N people (including themselves). With everyone wearing face masks and keeping their distance, it's almost impossible to recognize anyone else.

Alice counts A people standing in front of her, while Bob counts B people standing behind him.

Even if they can't see each other, they still want to know how far apart they are. Can you help them find out how many people are standing **between** Alice and Bob (excluding both of them)?

INPUT

One line contains three integers N, A, and B ($2 \leq N \leq 1000; 0 \leq A, B \leq N - 1; A + B \neq N - 1$) — the number of people in the queue, the number of people in front of Alice, and the number of people behind Bob, respectively.

OUTPUT

Output a single integer — the number of people standing between Alice and Bob.

EXAMPLE

INPUT	OUTPUT
`10 2 3`	`3`
`7 4 5`	`2`
`998 244 353`	`399`

Note

In the first example, the line, from front to back, can be described as `ooAoooBooo` where A is where Alice is and B is where Bob is.

In the second example, the line, from front to back, can be described as `oBooAoo` where A is where Alice is and B is where Bob is.