International Collegiate Programming Contest 2025
Asia Dhaka Regional

Hosted By
Bangladesh University of Business & Technology (BUBT)

Sponsored by

Platinum

Gold

Silver

General

Supported by

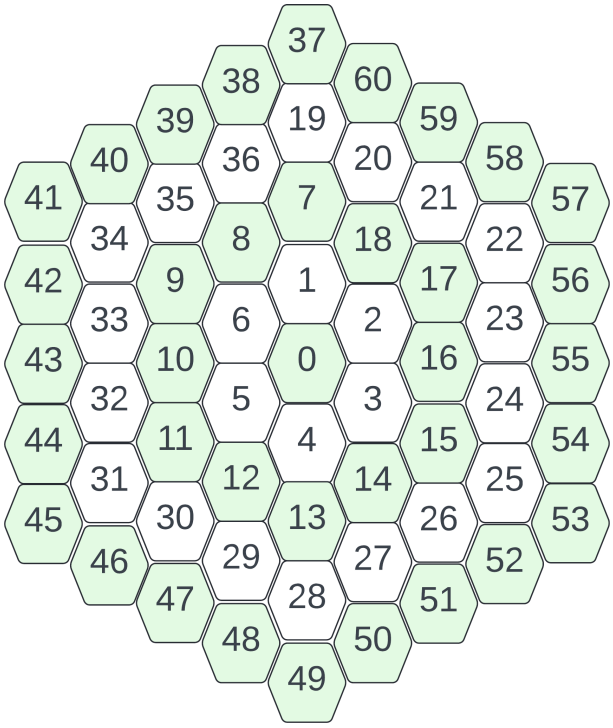
You get 10 problems, 18 pages and 300 minutes.

Problem A. Mission Hexa

You are an elite mercenary who has infiltrated the legendary beehive of the Queen Bee. The hive is built from regular hexagonal cells arranged in concentric layers around the queen's chamber at the center. The central cell is numbered 0.

The hive has $(n + 1)$ **layers** numbered from 0 to n. Layer 0 contains only the queen's chamber (cell 0). For every integer x $(1 \leq x \leq n)$, layer x is defined as the set of cells that are adjacent to at least one cell of layer $(x - 1)$ and are not part of any smaller layer. Each layer surrounds the previous (except layer 0) and is surrounded by the next (except layer n).

The **cells** are numbered layer by layer outward from the center. Within each layer, the numbering starts at the *topmost* cell of that layer. Numbering proceeds *clockwise* for odd-numbered layers and *counter-clockwise* for even-numbered layers.



You are currently at the queen's chamber at cell 0. Your **mission** is to kill the queen, and then escape the beehive from cell k.

Every cell other than the queen's chamber houses a **guard bee**. If a single guard bee anywhere in the hive is alive at the moment you exit the queen's chamber, you will be detected and killed immediately. Therefore, any bee that remains alive while you leave the center is fatal.

From the queen's chamber, you can use a powerful **laser gun** and fire any number of shots. Each shot is a *perfect straight ray* that originates at the center of the queen's chamber and extends to infinity in some direction. A shot kills a guard bee if and only if the center of that bee's cell lies *exactly* on the line of fire. The shot must be fully precise: if the center of a cell is missed by even the smallest amount, the guard bee in that cell remains alive.

After killing the queen and all the guards, you will walk from cell 0 to cell k along adjacent cells. A *path* is a sequence of cells where each consecutive pair of cells is adjacent. A path from one cell to another is considered *shortest* if and only if it contains the minimum possible number of cells among all paths connecting those two cells. Two paths are considered *different* if and only if there exists at least one position where the corresponding cells of the two paths are not the same.

Because you are in a hurry to watch a football match after the mission, you need to calculate the **minimum number of laser shots** you need to fire and the **number of distinct shortest paths** from the queen's chamber back to your escape cell. As the answers can be very large, output them **modulo** $(10^9 + 7)$.

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 space-separated integers n $(2 \le n \le 10^6)$ and k $(1 \le k \le 3n(n+1))$ — the number of outer layers and the escape cell.

Output

For each test case, output two space-separated integers in a line — the minimum number of laser shots you need to fire modulo $(10^9 + 7)$ and the number of different shortest paths from the queen's chamber to the escape cell modulo $(10^9 + 7)$.

Sample Input	Sample Output
3 2 9 4 20 1000000 3000000000000	12 1 36 3 781301591 1000000

Note

In the first test case, you can kill all the guards in 12 laser shots by directing them at equal angular intervals of 30°. It can be proven that this is the minimum number of shots required. The shortest path to the escape cell is $0 \to 6 \to 9$.

The beehive of the second test case is illustrated in the statement. The three distinct shortest paths are $0 \to 1 \to 7 \to 20$, $0 \to 1 \to 18 \to 20$, and $0 \to 2 \to 18 \to 20$.

Problem B. Boulevard of Broken Cars

The government of Ajobdesh launched an ambitious project to build a national highway, spanning the entire country, costing billions of dollars. But the construction company decided that high-quality bitumen was optional and spent most of the budget on million dollar pillows. As a result, the highway is riddled with potholes and cars break down all the time. As stranded vehicles become a common sight along the highway, the tow truck business is booming.

There are n tow truck companies operating on the highway. The i^{th} company operates a truck which is positioned at x_i and moves at speed v_i. If a car breaks down at position y, the time required to reach it for the i^{th} truck is $\frac{|y - x_i|}{v_i}$. You want to know the minimum response time for a tow truck if a car breaks down.

You must answer q queries. Each query will be one of the following two types.

- `+ x v` — A new company opens with a truck at position x with speed v.
- `? y` — A car breaks down at position y. You have to find the minimum time taken by any tow truck to reach the car.

Note that the queries of the second type are independent, you can assume that all tow trucks are at their initial positions before each query.

Input

The first line contains T ($1 \le T \le 10^4$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the number of tow truck companies. Each of the following n lines contain two integers x_i ($-10^{14} \le x_i \le 10^{14}$) and v_i ($1 \le v_i \le 10^9$).

The next line contains q ($1 \le q \le 3 \cdot 10^5$) — the number of queries. Each of the following q lines contain the description of the queries in the format described above ($-10^{14} \le x, y \le 10^{14}$ *and* $1 \le v \le 10^9$).

It is guaranteed that the sum of n over all cases does not exceed 10^5 and the sum of q over all cases does not exceed $3 \cdot 10^5$.

Output

For each query of the second type, output a single real number in a line — the minimum response time. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, if your answer is a, and the jury's answer is b, then your answer is accepted, if and only if $\frac{|a - b|}{max(1, b)} \le 10^{-6}$.

Sample Input	Sample Output
2 1 0 1 3 ? 1 + 2 2 ? 1 2 0 2 2 2 8 ? 1 ? -1 + 3 5 ? 1 ? -1 + 1 9 ? 1 ? -1	1.000000000000000 0.500000000000000 0.500000000000000 0.500000000000000 0.400000000000000 0.500000000000000 0.000000000000000 0.222222222222222

Note

In the first case, there is initially only one truck at $x = 0$ with speed $v = 1$.

- Query 1: A car breaks down at $y = 1$. It takes 1 second for the only truck to reach it.
- Query 2: A new truck appears at $x = 2$ with speed $v = 2$.
- Query 3: A car breaks down at $y = 1$. The initial truck at $x = 0$ can reach it in 1 second, but the new truck at $x = 2$ can reach it in 0.5 seconds.

Problem C. Gas Reservoir

Consider a three-dimensional grid of size $X \times Y \times Z$, representing underground layers. The (x, y) plane corresponds to the horizontal surface, while the z-axis represents depth below the surface, with $z = 0$ denoting the topmost layer.

Each cell of the grid is of exactly one of the following two types:

- **Rock**: denoted by the character '#'.
- **Gas**: denoted by the character '.', where each gas cell contains exactly 1 unit of gas.

A **reservoir** is defined as a maximal connected component of gas cells. Two gas cells are considered adjacent if and only if they share a common face. More formally, a gas cell with coordinates (x, y, z) is adjacent to another gas cell if their coordinates differ by exactly 1 in one of the three dimensions and are equal in the other two dimensions; that is, the adjacent coordinates are $(x \pm 1, y, z)$, $(x, y \pm 1, z)$, or $(x, y, z \pm 1)$.

You may dig a borehole from the surface. A **borehole** is defined by choosing a surface coordinate (x, y) and drilling vertically downward through all depths from the top layer $(z = 0)$ to the bottom layer $(z = Z - 1)$.

During drilling, the drill passes through both rock and gas cells. Whenever the drill encounters a gas cell, it immediately collects the entire reservoir containing that cell. Drilling continues downward and may intersect multiple reservoirs at different depths. Each reservoir is collected at most once, even if the drill intersects it multiple times within the same vertical column. Each gas cell contributes exactly 1 unit to the total collected gas.

Thus, drilling a borehole at surface coordinates (x, y) collects the total amount of gas contained in all distinct reservoirs that include at least one gas cell in the vertical column $(x, y, 0), (x, y, 1), \ldots, (x, y, Z - 1)$.

Your task is to determine the maximum total amount of gas that can be collected by drilling a single borehole at an optimal choice of surface coordinates (x, y).

Input

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

The first line of each test case is a blank line.
The second line of each test case contains three space-separated integers X, Y and Z $(1 \leq X, Y, Z \leq 50)$ — the dimensions of the grid.
The third line of each test case is a blank line.
Then follow Z layers, from top $(z = 0)$ to bottom $(z = Z - 1)$.

Each layer consists of X lines, each containing a string of length Y, consisting only '#' (rock) and '.' (gas).
Consecutive layers are separated by a blank line.

Output

For each test case, output a single integer in a line — the maximum total number of gas units that can be collected by drilling exactly one borehole.

Sample Input	Sample Output
2 2 2 3 .# #. .. .. ## #. 1 2 2 .# #.	7 1

Problem D. Save the Wonderland

In the magical land of Wonderland, there are N cities connected by roads. The road network has a special property — there is **exactly one path between any two cities**. A **path** is a sequence of different cities connected sequentially by roads.

The Queen of Wonderland maintains many teams of K **guards** in her barracks. She wants to deploy one of these teams to protect all cities against potential threats within the kingdom. Each guard protects the city they are stationed in, as well as all cities whose distance from that city along the roads is at most R. The **distance** between two cities is defined as the number of roads on the unique path connecting them.

All guards in a single team have the same protection radius R. Since the Queen wants to keep her strongest guards in reserve, she wishes to choose a team and deploy it in such a way that every city is protected while minimizing the radius R.

Your task is to determine the minimum possible value of R such that it is possible to place the K guards in some cities so that every city in Wonderland is protected by at least one guard.

Input

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

The first line of each test case contains two integers N and K $(1 \leq K \leq N \leq 10^5)$ — the number of cities in Wonderland and the number of guards respectively.

Each of the next $N-1$ lines contain two integers U and V $(1 \leq U, V \leq N, U \neq V)$ — indicating a road connecting city U and city V.

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

Output

For each test case, print a single integer in a line — the minimum possible value of R.

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

Problem E. Love Marriage

Shanto has finally decided that it's time for him to get married!

Although it took a while for him to reach this decision, it didn't take long for the news to spread like wildfire. Soon, Shanto's phone started to ring nonstop, with n girls requesting to chat with him, each seeing him as a perfect life partner. Being overwhelmed, Shanto devised an algorithm to find his **favorite girl**.

Here's his plan:

- Tomorrow, Shanto will chat with the girls one by one in the order they called.
- For every girl, Shanto will calculate the **happiness** he feels during the chat.
- At any point, Shanto will keep track of the favorite girl he's met so far.
- After chatting with the first girl, she's automatically the favorite girl (for now).
- For subsequent girls:
  - If the happiness Shanto feels chatting with her is less than that of the current favorite girl, he will stick with his favorite girl.
  - If it's greater, she will become the new favorite girl.
  - If it's equal, Shanto will **flip an unbiased coin**. If the coin lands on heads, he will stick to his current favorite girl. Otherwise, this girl will become the new favorite girl.
- If Shanto can finish chatting with all the girls, he will choose his favorite girl to become his wife.
- However, at any point, if Shanto chats with k consecutive girls and no girl **strictly surpasses** his maximum happiness, he'll cancel the rest of the dates and go to sleep. In this case, his current favorite girl gets to become "Mrs. Shanto"!

Being an honest boy, Shanto shares all the details about the algorithm with the girls. But here's a twist: Shanto himself doesn't know the value of k yet. Influenced by one of his roommates, Shanto has developed a habit of making important decisions at the last moment. So, he decided to sleep on it and choose k tomorrow morning.

Meanwhile, the girls, excited and anxious, started fantasizing about their chances. Some have tried to read Shanto's mind and guess the value of k. Others have decided to pray for an optimal k to maximize their chances. Now, they've turned to you, the only person who can predict Shanto's happiness levels, for help.

Your task is to calm their nerves by answering their queries.

Input

The first line of the input contains two space-separated integers n ($2 \le n \le 10^5$) and q ($1 \le q \le 2 \times 10^5$) — the number of girls who called Shanto and the number of queries.

The second line contains n space-separated integers a_1, a_2, ..., a_n — where a_i ($1 \le a_i \le 10^9$) is the predicted happiness Shanto will feel when chatting with the i-th girl.

Then q lines follow, each containing a query of one of the following two types:

- `"1 i k"`: the i-th girl ($1 \le i \le n$) asks about the probability of her becoming Shanto's wife for the given value of k ($1 \le k \le n$).
- `"2 i"`: the i-th girl ($1 \le i \le n$) asks about the optimal value of k that maximizes the probability of her becoming Shanto's wife and the corresponding probability.

Output

For each query, output the corresponding answer in a line.

For each query of type 1, output one integer $p\,(0 \leq p < 10^9 + 7)$.

For each query of type 2, output two space-separated integers $k\,(1 \leq k \leq n)$ and $p\,(0 \leq p < 10^9 + 7)$. If multiple values of k are optimal, you can output any of them.

Output all probabilities modulo $10^9 + 7$. Formally, if a probability can be expressed as an irreducible fraction $\frac{x}{y}$, you have to output $p = xy^{-1}\ mod\ (10^9 + 7)$, where y^{-1} is an integer such that $yy^{-1}\ mod\ (10^9 + 7) = 1$.

Sample Input	Sample Output
15 6 5 2 7 6 3 7 2 4 4 7 5 9 1 10 9 1 1 1 1 3 5 2 6 2 9 2 14 2 12	1 500000004 5 500000004 1 0 15 1 8 0

Note

In the sample test case, if $k = 5$, Shanto will chat with the first 8 girls and then go to sleep. Here, the 3rd girl and the 6th girl are valid candidates for being Shanto's favorite girl, each having a $\frac{1}{2}$ probability of becoming Shanto's wife.

Problem F. Morning Walk

Kabul Kabir is a computer science student who decided to improve his health by taking a regular morning walk. He walks on a park trail that forms a closed, simple loop of total length L meters. The trail has multiple entry points and walkers may move in either direction along the trail.

Kabul always walks in the forward direction at a constant speed of v_0 meters per second and continues walking for exactly T seconds.

While walking, Kabul encounters other walkers on the trail. Some walk in the same direction, while others walk in the opposite direction. Whenever Kabul and another walker occupy the same position on the trail at the same time, Kabul considers it a **meeting**. Meetings can occur either when they cross paths (opposite directions) or when one overtakes the other (same direction).

For each walker X_i, Kabul records the following information:

- t_i – the time (in seconds) after Kabul starts walking when he first meets X_i.
- v_i – the **signed** walking speed of X_i in meters per second: positive if the walker moves in the same direction as Kabul, and negative if the walker moves in the opposite direction.

Your task is to determine how many times in total Kabul meets each walker during the time interval from 0 to T, **including the first meeting and any meeting that occurs exactly at time** T.

Input

The first line of the input contains a single integer K ($1 \le K \le 10^4$) — the number of test cases.

The first line of each test case contains four space-separated integers L, v_0, T, and N ($1 \le L \le 10^6$, $1 \le v_0 \le 10^3$, $1 \le T \le 10^7$, $1 \le N \le 2 \cdot 10^5$) — the length of the trail in meters, Kabul's walking speed in meters per second, the total walking time in seconds, and the number of other walkers, respectively.

Each of the next N lines contains two space-separated integers t_i and v_i ($0 \le t_i \le T$, $-10^3 \le v_i \le 10^3$, $v_i \ne 0$, $v_i \ne v_0$) — the time of the first meeting with the i-th walker and that walker's signed walking speed in meters per second.

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

Output

For each test case, output a single line containing N space-separated integers. The i-th integer should be the total number of meetings Kabul has with the i-th walker, listed in the same order as the input.

Sample Input	Sample Output
3 200 30 75 1 45 10 400 5 5400 3 600 4 750 -6 1200 6 1000 5 5400 2 0 6 5400 -5	4 13 128 11 6 1

Note

In the first sample test case, the trail length is 200 meters and Kabul walks at 30 m/s for 75 seconds. The walker moves in the same direction at 10 m/s and first meets Kabul at $t = 45$. They meet again at $t = 55$, $t = 65$, and $t = 75$, resulting in a total of 4 meetings.

Problem G. Nonogram

Alice has picked up a new logic puzzle called a nonogram. A nonogram is played on a board consisting of n cells arranged in a row. You are given a sequence $b_1, b_2, \ldots, b_k$. You must color each cell black (`'1'`) or white (`'0'`) such that the contiguous blocks of black cells have lengths exactly $b_1, b_2, \ldots, b_k$.

For example, if $n = 6$ and the clue sequence is $[3, 1]$, the board must contain a block of 3 black cells, followed by at least one white cell, followed by a block of one black cell. There can be any number of white cells before the first block, after the last block, or between blocks (as long as there is at least one). So valid solutions are `011101`, `111001` or `111010`.

Alice has become a master at solving nonograms and now finds them boring. To make things more interesting, her friend Bob introduces an additional constraint. He colors some of the cells white (`'0'`), and leaves the rest empty (`'?'`). Alice must solve the nonogram while ensuring every cell Bob colored remains white.

Help Alice figure out if it is possible to find a solution to the puzzle consistent with Bob's constraints. In other words, you need to determine if you can color each `'?'` cell black or white so that the contiguous blocks of black cells have lengths exactly $b_1, b_2, \ldots, b_k$.

In addition, if it is solvable, then for each cell, you must find whether it is definitely black (`'1'`), definitely white (`'0'`), or can be either (`'?'`).

Input

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

Each case is described by three lines. The first line contains two integers n ($1 \le n \le 10^6$) and k ($0 \le k \le n$) — the length of the strip and the number of blocks. The second line contains a string of length n, consisting of characters `'0'` and `'?'` — which describes Bob's constraint. The third line contains k integers $b_1, b_2, \cdots, b_k$ ($1 \le b_i \le n$ *and* $\sum_{i=1}^{k} b_i \le n - k + 1$) — the lengths of the contiguous black blocks.

It is guaranteed that the sum of n over all cases does not exceed 2×10^6.

Output

For each test case,

- If no valid solution exists, print `No`.
- Otherwise, print `Yes` on the first line, followed by a string of length n on the second line where the i-th character is
  - `'1'` if the i-th cell is black in **every** valid solution.
  - `'0'` if the i-th cell is white in **every** valid solution.
  - `'?'` if the i-th cell is black in some solutions, but white in others.

Sample Input	Sample Output
5 5 2 ??00? 2 1 5 1 ???0? 2 5 1 ??0?? 3 4 0 ??0? 6 2 ?????? 3 1	Yes 11001 Yes ?1?00 No Yes 0000 Yes ?11???

Note

In the first case, the only solution consistent with Bob's constraints is `11001`.

In the second case, there are two valid solutions `01100` and `11000`. The first and the third character can be either '`0`' or '`1`' depending on the solution, so, they are marked as '`?`'. The second character is '`1`' in both solutions, whereas the last two characters are always '`0`'. Therefore, the resulting output is `?1?00`.

In the third case, there is no way to color 3 consecutive cells black without touching the third cell. Thus the puzzle has no valid solution.

Problem H. Optimal Balancing Strategy

You are given an array of positive integers A of size n. In one operation, you may perform the following:

- Choose any indices i, j such that $1 \leq i, j \leq n$,
- Choose a positive integer p that divides A_i, formally, $p \mid A_i$,
- Update $A_i := \frac{A_i}{p}$,
- Update $A_j := A_j \cdot p$.

Each of the operation has a cost of p.

After performing the operation any number of times (possibly zero), the final array must satisfy the following requirements, one from each of your three friends:

- Muhaimin wants at least a elements to be **evenly even**. A positive integer is called evenly even if it has no odd prime factor.
- Noshin wants at least b elements to be **oddly odd**. A positive integer is called oddly odd if it has no even prime factor.
- Osman wants the whole array to be **balanced**. An array is called balanced if each element of the array is either evenly even, or oddly odd, or both. For example, [4, 27] is a balanced array, while [6, 9] is not.

Determine the minimum total cost required to satisfy all three requirements. It can be proved that it is always possible to satisfy all three requirements.

Input

The first line of the input contains a single integer t ($1 \leq t \leq 500$) — the number of test cases.

The first line of each test case contains three space-separated integers n, a and b ($2 \leq n \leq 2 \cdot 10^5, 1 \leq a, b \leq (n-1), 2 \leq a+b \leq n$) — the size of the array, the minimum required number of evenly even elements and the minimum required number of oddly odd elements respectively.

The third line of each test case contains n integers $A_1, A_2, \ldots, A_n$ ($1 \leq A_i \leq 10^6$) — the elements of the array initially.

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 a single integer in a line — the minimum total cost required to make the array balanced with at least a evenly even and at least b oddly odd elements.

Sample Input	Sample Output
3 8 3 3 6 10 12 14 18 20 22 24 3 1 1 8 4 4 5 2 3 2 3 5 8 13	21 4 0

Note

In the second test case, we can choose $i = 2$, $j = 1$ and $p = 4$ (divide A_2 by 4 and multiply A_1 by 4), resulting in the array $[32, 1, 4]$.

Other possible final arrays with the same minimum cost are:

- $[8, 1, 16]$
- $[32, 4, 1]$
- $[8, 16, 1]$

Problem I. Two Strings Attached

Pebae has a string A and Mr Blue has a string B, both of length N (1-indexed). For a string S, let $S[i \dots j]$ denote the contiguous substring from index i to j (inclusive).

Now they want to find all the ways their strings can be attached! Specifically, count the number of pairs (i, j) with $1 \le i, j \le N$ such that:

$$A[1 \dots i] + B[j \dots n] = A[j \dots n] + B[1 \dots i]$$

.

Here, + denotes string concatenation. Please help them find the number of valid pairs.

Input

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

The first line of each test case contains an integer N ($1 \le N \le 3 \times 10^5$) — the length of both strings.

The second line contains a string A of length N consisting of lowercase English letters.

The third line contains a string B of length N consisting of lowercase English letters.

It is guaranteed that the sum of N over all test cases does not exceed 3×10^6.

Output

For each test case, output a single integer — the number of valid pairs.

Sample Input	Sample Output
2 3 aba bab 6 ababca ababca	4 2

Note

In the first test case, $A =$ `aba` and $B =$ `bab`. The valid pairs (i, j) are:

- $(1, 1)$: $A[1 \dots 1] + B[1 \dots 3] =$ `a` + `bab` = `abab` and $A[1 \dots 3] + B[1 \dots 1] =$ `aba` + `b` = `abab`
- $(1, 3)$: $A[1 \dots 1] + B[3 \dots 3] =$ `a` + `b` = `ab` and $A[3 \dots 3] + B[1 \dots 1] =$ `a` + `b` = `ab`
- $(3, 1)$: $A[1 \dots 3] + B[1 \dots 3] =$ `aba` + `bab` = `ababab` and $A[1 \dots 3] + B[1 \dots 3] =$ `aba` + `bab` = `ababab`
- $(3, 3)$: $A[1 \dots 3] + B[3 \dots 3] =$ `aba` + `b` = `abab` and $A[3 \dots 3] + B[1 \dots 3] =$ `a` + `bab` = `abab`

Problem J. C-Style String Length

You are analyzing a string literal written in the **C programming language**. In C, special characters inside string literals are represented using **escape sequences**, all of which begin with a backslash (\).

Some relevant escape sequences are:

- \\: represents a single backslash character \
- \0: represents the null character (NUL). In C, the function `strlen` stops counting characters when it encounters the first NUL.

You are given a string S, which represents the characters written **inside the quotation marks** of a C string literal. The string S consists only of two characters: backslash (\) and zero (0).

Your task is to determine the value that the C function `strlen` would return **after interpreting all escape sequences in** S. If S contains an incomplete escape sequence (i.e., a single backslash at the end of the string), then S is considered **INVALID**, and you should report that instead.

The string S is scanned from **left to right** according to the following rules:

- If the current character is 0 and it is **not part of an escape sequence**, it is treated as a normal character and contributes 1 to the string length.
- If the current characters form the escape sequence \\, they produce a single backslash character and contribute 1 to the string length.
- If the current characters form the escape sequence \0, they produce a NUL character. At this point, `strlen` stops immediately, and the NUL character is **not counted**.
- If any escape sequence is incomplete, and the string is considered **INVALID**.

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 string S ($1 \le |S| \le 10^5$), containing the characters \ and 0 — the string literal inside the quotation marks passed as a parameter to `strlen`.

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

Output

For each test case, output a single integer in a line — the value returned by `strlen` after decoding the string. If the string is not valid, output `INVALID` instead. Note that the output is case-sensitive.

Sample Input	Sample Output
4 \\\\ \\0 \0\\00 \\\	2 2 0 INVALID

Note

In the first test case, \\\\ decodes to \\, so the length is 2.
In the second test case, \\0 decodes to \ followed by 0, so the length is 2.
In the third test case, \0\\00 begins with a NUL character, so `strlen` returns 0.
In the fourth test case, \\\ ends with an incomplete escape sequence, so it is **INVALID**.