Problem A
Angry Birbs

Everyone knows that green pigs are the enemies of the birbs because they are very greedy and often steal eggs from the birbs. A battle between birbs and pigs broke out to reclaim the eggs. The pigs had built fortresses and hidden the eggs inside. But with their strong physical abilities, the birbs chose to fly and dove straight into the pig's fortresses. It might seem like a mindless strategy, but it was very effective and completely destroyed the fortresses of the opponents and the eggs were retrieved countless times. Whether the fortress is on the ground, in the clouds, or underwater, the birbs can reach the pig's fortress.

With advancing technology and the huge collection of data on how the birbs attack, the pigs have concluded that the flight trajectory of the birbs can be represented by the equation:

$$h = a + b \cdot x + c \cdot x^2$$

where h is the current height above sea level (can be negative when the birb is underground or underwater), x is the current horizontal position of the birb, and a, b, and c are the parameters of the trajectory equation.

With this discovery, the pigs further study the characteristics of the birb's flight trajectory to come up with the best defense plan. However, the next wave of birb attacks is imminent, so the pigs will not have much time. Therefore, the pigs only choose to reinforce the roof or the floor in the following cases:

1. If the initial height of the birb increases to a maximum height and then decreases, we say that the birb's trajectory is a *curve down*. In this case, the pigs will reinforce the roof.
2. If the initial height of the birb decreases to a minimum height and then increases, we say that the birb's trajectory is a *curve up*. In this case, the pigs will reinforce the floor.
3. If the birb's trajectory does not have a highest or lowest point, we say that the birb's trajectory is *not curved*. In this case, the pigs will be confused because there might be something wrong with their finding.

Given the parameters a, b, and c of the birb's trajectory equation, help the pigs determine which type of trajectory it is among the three types described above.

Input

The first line contains an integer t ($1 \le t \le 10\,000$) – the number of test cases. The description of each test case is as follows.

The first and only line of each test case contains three integers a, b, and c ($|a|, |b|, |c| \le 10^9$) – the parameters of the trajectory equation of a birb about to dive into the pig's fortress.

Output

For each test case, print a line:

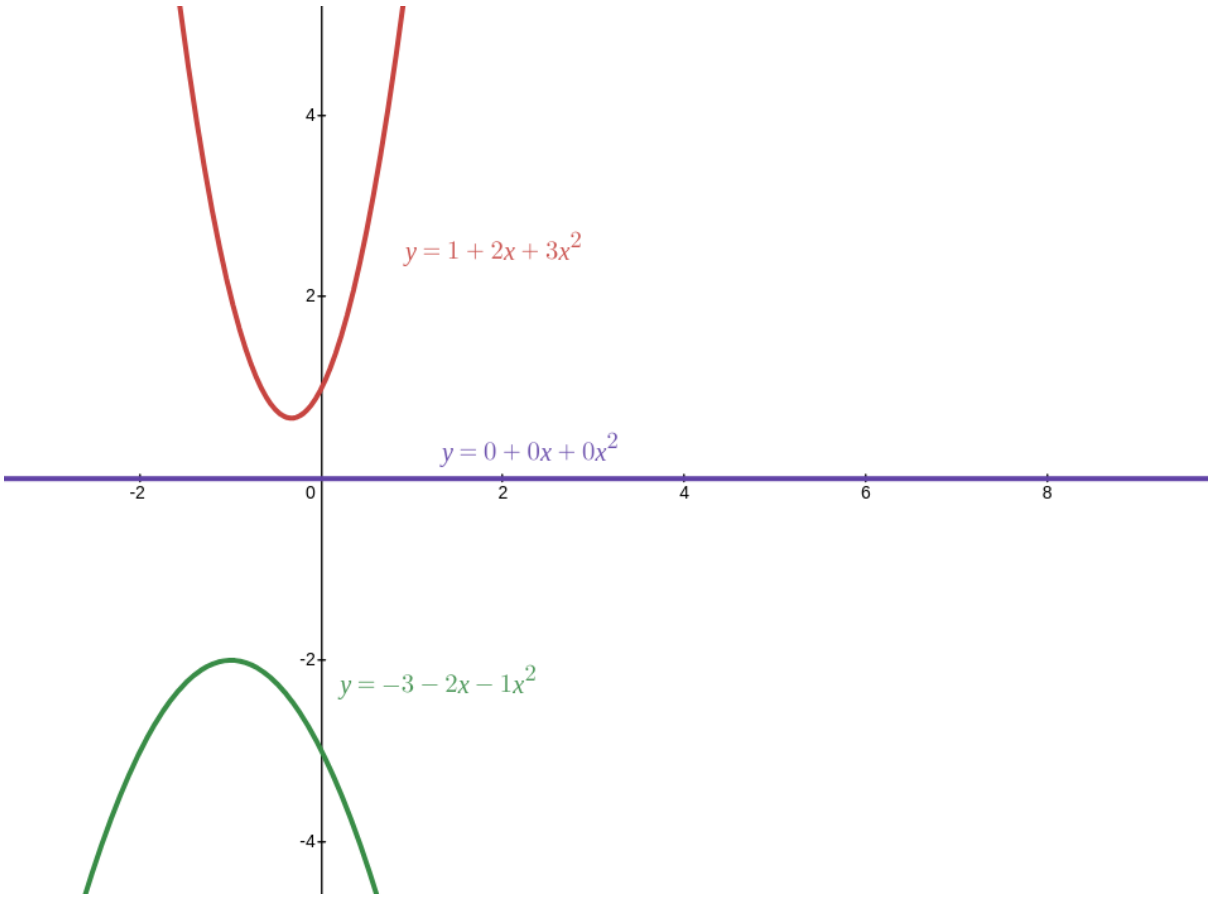
- If the birb's trajectory is a *curve down*, print "`CURVE DOWN`" (without the quotation marks).

- If the birb's trajectory is a *curve up*, print "`CURVE UP`" (without the quotation marks).
- If the birb's trajectory is *not curved*, print "`NO CURVE`" (without the quotation marks).

Sample explanation

The below image illustrates the birbs' trajectory in the sample.

- The red line is the birb's trajectory from the first test case.
- The green line is the birb's trajectory from the second test case.



Sample Input 1	Sample Output 1
2 1 2 3 -3 -2 -1	CURVE UP CURVE DOWN

Problem B
Between Strings

Today, in an algorithms course, Hanh learns lexicographical order.

The professor gives Hanh a counting task: *Given two strings a and b and an integer l, count the number of strings c of length l such that c is greater than a and less than b, in terms of lexicographical order.*

This task is quite a simple one for Hanh. He manages to write the below function in a matter of second: `countBetween(a, b, l, p)`. This function works as below:

- It has four parameters: `a` and `b` are two strings of lowercase English letters, `l` and `p` are two positive integers.
- It correctly counts the number of strings c containing exactly `l` lowercase English letters, satisfying $a < c < b$ in terms of lexicographical order.
- It returns a non-negative integer which is the remainder after dividing the number of such strings c by `p`.

To make thing more interesting and challenging for Hanh, his professor now asks him to proceed multiple queries of counting strings between. In order to solve this harder problem, Hanh writes the following pseudo code:

```
function solve(s, w, d, p):
    a = "icpcvn"
    b = "icpcvn"
    res = 0

    for each i from 0 to length(s) - 1:
        if w[i] equals 'A':
            insert character s[i] at the end of string a
        else:
            insert character s[i] at the end of string b

        sum = 0
        for each integer l in d:
            sum += countBetween(a, b, l, p)

        res = res xor (sum % p * (i + 1))

    return res
```

The function `solve` takes four parameters:

- Two strings `s` and `w` **of the same length**, `s` contains lowercase English characters only, `w` contains characters `A` and `B` only.
- `d` is a sequence of positive integers.

- `p` is a positive integer.

In the above pseudo code, `length(s)` denotes the number of characters of string `s`, while `s[i]` denotes the `i`-th character of string `s`. Characters are numbered starting from 0.

Sadly, Hanh realizes that his code may run too slow with some of the professor's test data. Hence, he asks you to write a program to efficiently calculate the returning value of the above function, given the values of all its parameters.

As a reminder, a string $X = x_0x_1x_2 \dots x_m$ is considered *lexicographically less* than a string $Y = y_0y_1y_2 \dots y_n$ iff either of the below conditions hold:

- $m < n$ and $x_i = y_i$ for every $0 \le i \le m$
- There exists an index i such that $i \le \min(m, n)$, $x_i < y_i$ and $x_j = y_j$ for every $0 \le j < i$.

A string X is considered *lexicographically greater* than a string Y iff Y is *lexicographically less* than X.

Input

- The first line contains three integers q, n and p ($1 \le q \le 2 \cdot 10^6, 1 \le n \le 7 \cdot 10^4, 1 \le p \le 10^9$) — the length of parameters `s` and `w`, the number of elements of parameter `d` and the value of parameter `p`, respectively.
- The second line contains n integers $d_1, d_2, \dots, d_n$ ($1 \le d_i \le 10^9$) — the elements of parameter `d`.
- The third line contains a string of exactly q lowercase English characters — the parameter `s`.
- The forth line contains a string of exactly q characters `A` and `B` — the parameter `w`.

Output

Print a single integer — the returned value of the above function `solve`.

Sample Explanation

In the above example, `s = "ca"`, `w = "BA"`, `d = [6, 7, 8]` and `p = 45`. In the beginning, `a = b = "icpcvn"`.

When `i = 0`, `a = "icpcvn"` and `b = "icpcvnc"`.

- `countBetween("icpcvn", "icpcvnc", 6, 45)` returns 0.
- `countBetween("icpcvn", "icpcvnc", 7, 45)` returns 2 as there are 2 strings of length 7 between `icpcvn` and `icpcvnc`: `icpcvna` and `icpcvnb`.
- `countBetween("icpcvn", "icpcvnc", 8, 45)` returns $52 \mod 45 = 7$ as there are 52 strings of length 8 between `icpcvn` and `icpcvnc`: `icpcvnaa`, `icpcvnab`, ..., `icpcvnaz`, `icpcvnba`, `icpcvnbb`, ..., `icpcvnbz`.

When `i = 1`, `a = "icpcvna"` and `b = "icpcvnc"`.

- `countBetween("icpcvna", "icpcvnc", 6, 45)` returns 0.
- `countBetween("icpcvna", "icpcvnc", 7, 45)` returns 1 as there is 1 string of length 7 between `icpcvna` and `icpcvnc`: `icpcvnb`.
- `countBetween("icpcvna", "icpcvnc", 8, 45)` returns $52 \mod 45 = 7$ as there are 52 strings of length 8 between `icpcvna` and `icpcvnc`: `icpcvnaa`, `icpcvnab`, ..., `icpcvnaz`, `icpcvnba`, `icpcvnbb`, ..., `icpcvnbz`.

Sample Input 1	Sample Output 1
2 3 45 6 7 8 ca BA	25

This page is intentionally left blank.

Problem C
Chess Sudoku

In her free time, Chikapu likes playing Sudoku — a logic puzzle played on a 9×9 grid divided into 9 regions of size 3×3 (see figure C.1). In each cell, the player must write a digit between 1 and 9 (inclusive) such that each row, each column, and each of the nine 3×3 regions must contain all of the digits from 1 to 9.

1	2	3	4	5	6	7	8	9
8	4	6						
9	5	7						
3								
4								
5								
6								
7								
2								

Figure C.1: A partially filled Sudoku board

Chikapu also likes chess. She designed a new variant of the traditional Sudoku puzzle, which she has aptly named **Chess Sudoku**. In addition to the regular Sudoku rules, a valid **Chess Sudoku** board must satisfy the following requirements:

- If a chess knight can move from cell (x_u, y_u) to cell (x_v, y_v) in one step, the digits in these two cells must differ.
- Similarly, if a chess king can move from cell (x_u, y_u) to cell (x_v, y_v) in one step, the digits in these two cells must also differ.

Chikapu has a **Chess Sudoku** board, where she has written some digits on it. Your task is to fill in the empty cells to construct a valid **Chess Sudoku** board.

Below are some invalid **Chess Sudoku** boards:

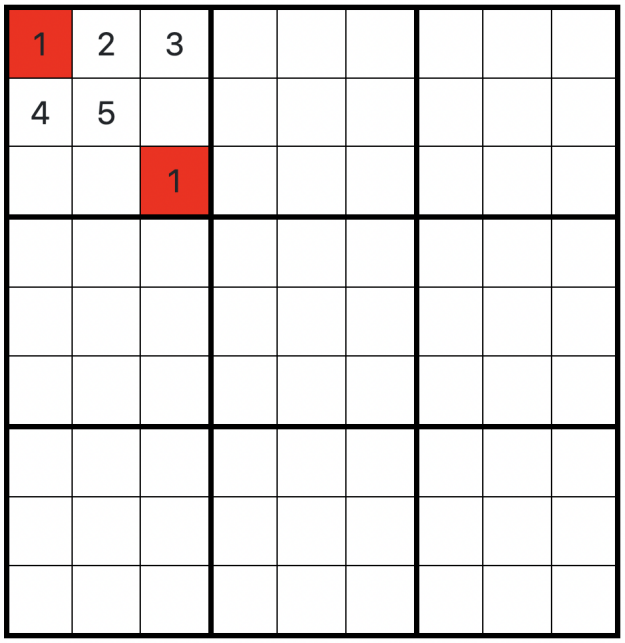


Figure C.2: The cells $(1, 1)$ and $(3, 3)$ belong to the same 3×3 region, but the numbers in the 2 cells are the same.

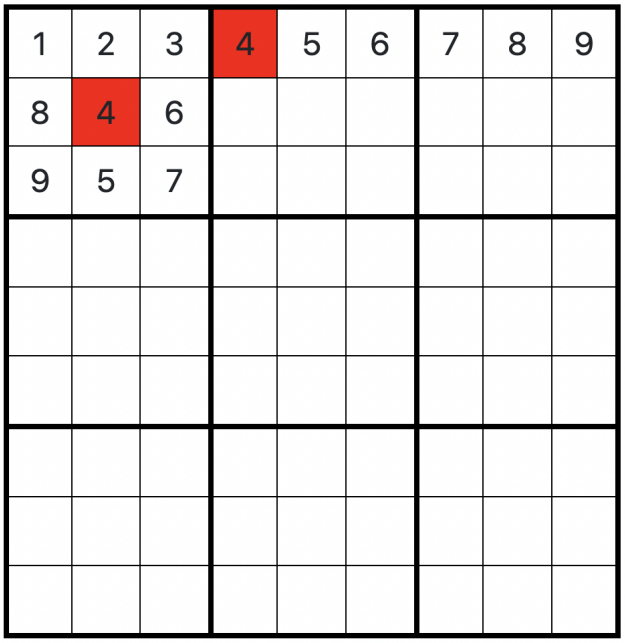


Figure C.3: A chess knight can move from cell $(1, 4)$ to cell $(2, 2)$, but the numbers in these 2 cells are the same.

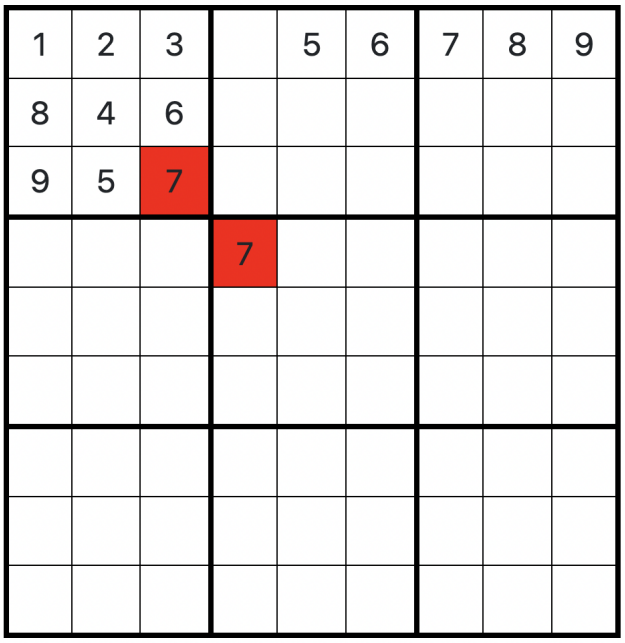


Figure C.4: A chess king can move from cell $(3, 3)$ to cell $(4, 4)$, but the numbers in these 2 cells are the same.

As a reminder, let (i, j) be the cell on the i-th row and the j-th column,

- a chess knight can move from cell (x_u, y_u) to cell (x_v, y_v) in one step iff $(x_u - x_v)^2 + (y_u - y_v)^2 = 5$,
- a chess king can move from cell (x_u, y_u) to cell (x_v, y_v) in one step iff $\max((x_u - x_v)^2, (y_u - y_v)^2) = 1$.

Input

The input consists of 9 lines, each contains exactly 9 digits, representing the given **Chess Sudoku** board. Empty cells are represented by 0s. There are at most 2 non-zero digits, which represent prefilled cells.

Output

If it is not possible to create a valid **Chess Sudoku** board with these prefilled cells, print a single line: `NO SOLUTION`.

Otherwise, print exactly 9 lines, each with exactly 9 digits, representing a valid **Chess Sudoku** board. If there are multiple solutions, you can output any of them.

Sample explanation

In the first sample, two prefilled cells are in the same column, but they are both filled with digit 1. This violates the rules of a valid **Sudoku** board. Hence it is not possible to construct a valid **Chess Sudoku** board.

Sample Input 1	Sample Output 1
100000000 100000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000	NO SOLUTION

Sample Input 2	Sample Output 2
000000000 000000000 000000000 000600000 000000000 000006000 000000000 000000000 000000000	NO SOLUTION

Sample Input 3	Sample Output 3
000000000 000000000 000000000 000000000 000000000 000000000 000000700 000007000 000000000	NO SOLUTION

Problem D
Dazzling Card Set

Slay the Spire is a popular video game that combines elements of roguelike and deck-building genres. It is a tactical game in which players navigate through a series of levels, fight enemies, and collect cards to build their deck. The goal is to climb to the top of the tower by defeating powerful bosses along the way. Each playthrough is unique as players encounter different cards, relics, and events that shape their strategy. *Slay the Spire* has received praise for its challenging gameplay, deep tactics, and replayability.

In the current playthrough, Lokk has a hand of n attack cards. The i-th card has a damage value of a_i. Lokk is playing the latest version of *Slay the Spire* and has acquired a new relic called *Radiance*. *Radiance* has the following effect: if in this turn, the cards you play are *Dazzling*, then after this turn, the *Vulnerable* status will be applied to all enemies.

The cards played are called *Dazzling* if the played cards are adjacent in the hand and no two played cards have the same damage value. In other words, if there exists two indices l and r ($1 \le l \le r \le n$), such that:

- all cards from position l to r are played in the current turn,
- no other card is played, and
- $a_i \neq a_j$ for all $l \le i < j \le r$.

then the played cards are called *Dazzling*.

With *Radiance* in hand, Lokk wants to use this relic as effectively as possible. Given the list of damage values of Lokk's cards in hand in the order they appear in the hand, help Lokk count the number of possible plays in the current turn for *Radiance* to take effect. Two plays are considered different if there exists an index i such that card i is played in one play but not in the other.

Input

The first line contains a positive integer t — the number of test cases. The description of each test case follows.

The first line of each test case contains an integer n ($1 \le n \le 200\,000$) — the number of cards in Lokk's hand.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the damage values of Lokk's cards in hand.

It is guaranteed that the sum of n over all test cases does not exceed $500\,000$.

Output

For each test case, print a single integer, the number of possible plays for the *Radiance* to take effect after this turn.

Sample explanation

In the first example, Lokk has a hand of cards with damage values $[4, 1, 1, 6, 2, 2]$. There are 10 ways to choose the *Dazzling* card sequences, as shown by the underlined card sequences below:

$[\underline{4}, 1, 1, 6, 2, 2]$	$[4, \underline{1}, 1, 6, 2, 2]$	$[4, 1, \underline{1}, 6, 2, 2]$	$[4, 1, 1, \underline{6}, 2, 2]$	$[4, 1, 1, 6, \underline{2}, 2]$
$[4, 1, 1, 6, 2, \underline{2}]$	$[\underline{4, 1}, 1, 6, 2, 2]$	$[4, 1, \underline{1, 6}, 2, 2]$	$[4, 1, 1, \underline{6, 2}, 2]$	$[4, 1, \underline{1, 6, 2}, 2]$

The card sequence $[1, 1, 6]$ is not considered *Dazzling*, because it contains two cards with damage points of 1.

In the second example, it is not possible to choose a *Dazzling* card sequence with multiple cards.

In the third example, in addition to the card sequences consisting of only one card, the card sequences consisting of two consecutive cards are also considered *Dazzling*.

Sample Input 1	Sample Output 1
3 6 4 1 1 6 2 2 3 2 2 2 9 6 9 6 9 6 9 6 9 6	10 3 17

Problem E
Extended Ping Pong

Ping pong is a simple but captivating sport with a clear scoring system. Matches are usually played in the form of 5 or 7 games, in which each game is won by the player who reaches 11 points first. However, if the score reaches 10-10, the match will continue until one player has a two-point advantage. Players take turns serving, with the serve changing after every two points. A point is awarded when the opponent fails to return the ball. The player or team that reaches 11 points with at least a two-point lead will win, and the overall winner of the match is determined by winning the majority of the games.

In this problem, we will consider ping pong matches with a slightly different scoring system. We will say that the winning player must have at least n points and be leading by at least two points. Alice and Bob are currently playing ping pong with each other using this new scoring system. After the match is over, Alice has scored s_A points, and Bob has scored s_B points. The match is intense and exciting, but after the match, both Alice and Bob are exhausted and have forgotten the details of the match. After resting, they want to remember together how the match unfolded. However, the number of possible scenarios that could result in the scores s_A and s_B is actually quite large.

Given three numbers n, s_A, and s_B, help Alice and Bob count the number of possible scenarios that could occur when they play ping pong with the new scoring system, so that after the match, Alice has s_A points and Bob has s_B points.

A ping pong scenario is defined as a sequence of events that occur during the match, where the i-th event records the player who scores after the i-th serve, and the winner is determined after the last event. For example, with $n = 3$:

- The scenario "Alice, Alice, Bob, Alice" is a valid scenario, with Alice scoring 3 points and Bob scoring 1 point.
- The scenario "Alice, Bob, Alice, Bob, Alice" is not a valid scenario. After the last event, Alice has 3 points, but Bob has 2 points, so the match is not over because Alice does not have a two-point lead over Bob.
- The scenario "Alice, Bob, Bob, Alice, Alice, Alice" is also a valid scenario, with scores $s_A = 4$ and $s_B = 2$. Now Alice is leading Bob by 2 points.

Two scenarios with the same final score are considered different if there exists an event i where the scoring player in one scenario is Alice while the scoring player in the other scenario is Bob.

Input

The first line contains a single integer T ($1 \leq T \leq 125$) – the number of test cases.

T test cases follow, each contains a single line with three integers n, s_A, and s_B ($2 \leq n \leq 10$, $0 \leq s_A, s_B \leq 10$) – respectively, the minimum score limit that the winning player needs to achieve, the final score of Alice after the match, and the final score of Bob after the match.

Output

For each test case, print a single integer, the number of possible scenarios when Alice and Bob play ping pong with the new scoring rule, so that after the match ends, Alice has s_A points, and Bob has s_B points.

Sample Input 1	Sample Output 1
2 2 0 0 2 0 2	0 1

Problem F
Finding RPS Strategy

Rock-paper-scissors is a game that everyone knows. A game consists of at least two players. Each player simultaneously chooses one of three shapes: rock, paper, or scissors. Rock beats scissors, scissors beats paper, and paper beats rock. If two players choose the same shape, it's a tie and neither wins nor loses.

Loc has just developed a rock-paper-scissors game on the computer and now he wants to test his game. Loc and the computer will play n rounds of rock-paper-scissors. Knowing the algorithm that the computer will use, Loc can predict the shape that the computer will choose for all n rounds. However, after working on the game all day, Loc is very tired. He doesn't want to make too many decisions during the test, so he will choose a number $k > 1$, and for every k consecutive rounds, Loc will choose only one shape. Specifically:

- In rounds $1, 2, \ldots, k$, Loc will choose the same shape.
- In rounds $k + 1, k + 2, \ldots, 2 \cdot k$, Loc will choose the same shape.
- $\ldots$
- In the last $(n \bmod k)$ rounds, Loc will choose the same shape.

Given the list of all shapes that the computer will choose, help Loc choose the number $k > 1$ such that the total number of rounds that Loc can win is **maximized**. If there are multiple possible values of k, find the **largest** possible value of k so that Loc does not need to make a lot of decisions.

Input

The first line contains an integer t ($1 \le t \le 10\,000$) – the number of test cases. The description of each test case is as follows.

The first and only line of each test case consists of a string s ($2 \le |s| \le 200\,000$, $s_i \in \{\texttt{"R"}, \texttt{"P"}, \texttt{"S"}\}$) – the shapes that the computer will choose in n rounds.

- If $s_i = \texttt{"R"}$, the computer will choose rock in round i;
- If $s_i = \texttt{"P"}$, the computer will choose paper in round i;
- If $s_i = \texttt{"S"}$, the computer will choose scissors in round i.

It is guaranteed that the total length of the given strings in all test cases does not exceed $200\,000$.

Output

For each test case, print a single integer which is the value of $k > 1$ that helps Loc win as many rounds as possible. If there are multiple valid answers, print the **largest** answer.

Sample explanation

In the first example, the computer will have the sequence of shapes as "RRRPPS". The sequence of shapes that Loc can have is "PPPSSS".

Player	Sequence of shapes	Number of winning rounds
Computer	R R R P P S	0
Loc	P P P S S S	5

In the second example, Loc can choose $k = 8$ with the sequence of shapes "PPPPPPPPR"

Player	Sequence of shapes	Number of winning rounds
Computer	R R S R R P R R S	1
Loc	P P P P P P P P R	7

Loc can also choose $k = 2$ with the sequence of shapes "PPRRSSPPR" and still win 7 rounds. However, $k = 2$ is not the maximum answer found.

Sample Input 1	Sample Output 1
3 RRRPPS RRSRRPRRS RPSRPSRPS	3 8 4

Problem G
Guarded Rook Combinations

Chess is a strategic board game that has been played for centuries. It involves two players taking turns moving their pieces on a square grid board. The objective of the game is to capture the opponent's king in a position where it cannot escape, known as "checkmate". One of the six different types of chess pieces is the rook. The rook is a powerful piece that can move horizontally or vertically any number of squares, as long as there are no obstacles in its path. It is usually represented by a structure resembling a tower and is considered one of the most valuable pieces in the game due to its ability to control multiple lines and contribute to various tactics and strategies.

As a passionate chess enthusiast, MofK always spends time to study chess pieces, positions, playing strategies, and then provides his own evaluations and comments to use in chess matches with Grandmasters in major tournaments. Today, MofK decides to dedicate an entire day to studying the properties of rooks. MofK has many chess boards and chess sets, so he decides to conduct the study with an $n \times n$ board instead of an 8×8 chesisboard. Each square on the chessboard can have at most one rook placed on it. Initially, MofK has placed some rooks on certain squares of his board. The more rooks he places, the more pairs of rooks that *guard* each other he sees in the chess position. To improve his predictive ability, MofK wants to add **exactly** k more rooks to the board, such that there are **exactly** p pairs of *guarding* rooks.

Two different rooks on the board are considered *guarding* each other if:

- the two rooks are in the same row or the same column on the board, and
- there is no other rook between the two rooks.

Given the initial board, the numbers k and p, help MofK find a way to add **exactly** k rooks to the chessboard such that there are **exactly** p pairs of *guarding* rooks, or indicate that there is no way to achieve that.

Input

The input contains multiple test cases. Each test case is presented as below:

- The first line contains three integers n, k and p ($1 \le n \le 10^3, 0 \le k \le 10^6, 0 \le p \le 10^9$) — the size of the board, the number of rooks to be added, and the number of pairs of *guarding* rooks that should appear on the board, respectively.
- The last n lines describe the initial board, each contains a string of length n. Each character of each string is either `"R"` (which represents a cell with a rook) or `"."` (which represents a cell without any rooks).

The input is terminated by a line containing three 0s which does not represent a test case. It is guaranteed that the sum of n^2 in all test cases does not exceed 2000^2.

Output

For each test case, if there is no way to add **exactly** k rooks to the board such that there are **exactly** p pairs of *guarding* rooks, print a single line `"NO"` (without quotes).

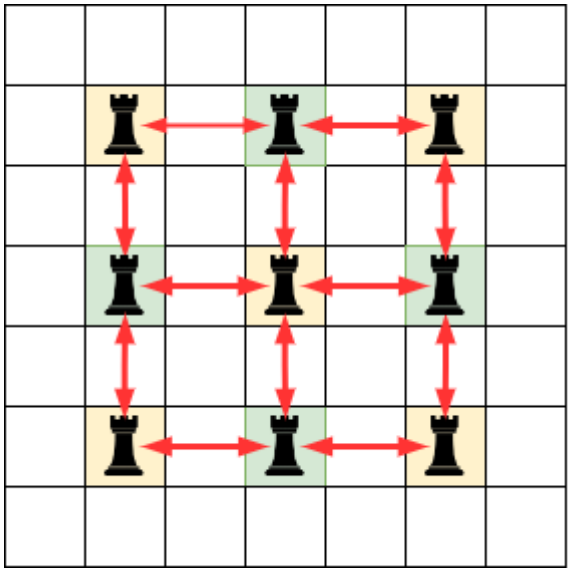
Otherwise, print "YES" (without quotes) on the first line. The next n lines describe the board after adding rooks in the same format as the input. Each line contains a string of length n where each character is either "R" or ".". "R" represents a cell with a rook, while "." represents a cell without any rooks.

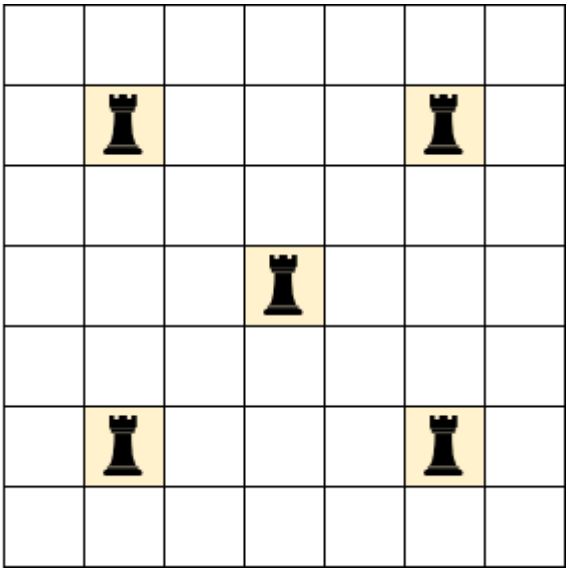
If there are multiple valid solutions, you can output any of them.

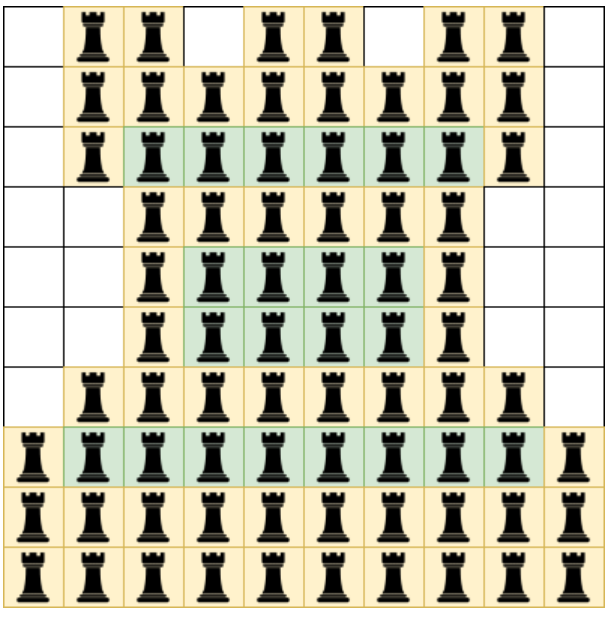
Sample explanation

These below figures demonstrate three test cases in the sample. The top-left one is the board after adding rook in the first test case, with each pair of *guarding* rooks is represented by a red line connecting the two rooks. The top-right one is the initial board in the second test case. The bottom one is the board after adding the rook in the third test case.

In the second test case, you can not add any rooks ($k = 0$), but initially there are only 4 pairs of *guarding* rooks. Hence it is not possible to have **exactly** $p = 12$ pairs.







Sample Input 1	Sample Output 1
7 4 12 .... .R...R. ....R... .... .R...R. .... 7 0 12 .... .R...R. ....R... .... .R...R. .... 10 22 136 .RR.RR.RR. .RRRRRRR. .R.....R. ..RRRRR.. ..R....R.. ..R....R.. .RRRRRRR. R........R RRRRRRRRR RRRRRRRRR 0 0 0	YES .... .R.R.R. .... .R.R.R. .... .R.R.R. .... NO YES .RR.RR.RR. .RRRRRRR. .RRRRRRR. ..RRRRR.. ..RRRRR.. ..RRRRR.. .RRRRRRR. RRRRRRRRR RRRRRRRRR RRRRRRRRR

This page is intentionally left blank.

Problem H
Hard Queries

Lokk loves data structure, and his favorite leisure activity is inventing new and exotic data structure problems, and challenge his friends to solve it. Today, Lokk came up with the problem below and sent it to Quang.

You are given an array a containing n elements $a_1, a_2, \ldots, a_n$. You task is to process q queries of the following types:

- $1\ p$: Swap two elements a_p and a_{p+1}.
- $2\ l\ r\ x$: Calculate the sum of square of all positions from l to r whose element is equal to x. Formally, calculate $\sum_{i=l}^{r} i^2 \times [a_i = x]$, where $[P]$ denotes an expression that evaluate to 1 if P is true, and 0 otherwise.
- $3\ l\ r\ x$: Define b as the array of all positions in array a whose element is equal to x, listed from 1 to n. Calculate $\sum_{i=l}^{r} {b_i}^2$.

Being a brilliant problem solver, Quang tackled the problem quickly. Therefore, Lokk put an additional challenge: Quang has to answers the queries **online**, meaning that the queries are encoded in a way that the correct answer to the current query is required to decode the next one.

This is too much for Quang to handle, so he gave up on the problem and ask the contestants to help him. Can you solve it?

Input

The first line contains two integers n and q ($2 \le n \le 2 \times 10^5$, $1 \le q \le 2 \times 10^5$) — the length of a, and the number of queries to process.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of array a.

Each of the next q lines describe a query to process. The first number is t ($1 \le t \le 3$) — the type of the query. The given queries will be encoded in the following way: let $last$ be the answer to the last query of the type 2 or 3 that you have answered (initially, $last = 0$).

If $t = 1$, an integer p' follows ($1 \le p' < n$). Perform the operation $p = (p' + last - 1) \bmod (n - 1) + 1$ to decode the query.

If $t = 2$ or $t = 3$, three integers l', r', and x' follow ($1 \le l', r', x' \le n$). Perform the following operations to decode the query:

- $l = (l' + last - 1) \bmod n + 1$
- $r = (r' + last - 1) \bmod n + 1$
- $x = (x' + last - 1) \bmod n + 1$
- If $l > r$, swap l and r.

For $t = 3$, it is guaranteed that r will not exceed the length of array b (defined in the query description).

Output

For each query of type 2 or 3, print a single integer — the answer to the query.

Sample explanation

Query 1 after decoding is 2 1 6 1. The positions from $l = 1$ to $r = 6$ whose element is equal to $x = 1$ are 1, 3 and 6. Therefore, the answer to this query is $1^2 + 3^2 + 6^2 = 46$.

Query 2 after decoding is 3 2 3 1. In this case, array b is $[1, 3, 6]$, so the answer to this query is $3^2 + 6^2 = 45$.

Query 3 after decoding is 2 6 6 2. Since $a_6 = 1$ which is different from $x = 2$, the answer to this query is 0.

Query 4 after decoding is 2 1 7 5. Since none of the elements of a equal $x = 5$, the answer to this query is 0.

Query 5 after decoding is 1 6. After swapping a_p and a_{p+1} (which is a_6 and a_7), array a become $[1, 2, 1, 3, 2, 2, 1]$.

Query 6 after decoding is 2 1 6 1. The answer to this query is $1^2 + 3^2 = 10$.

Query 7 after decoding is 3 2 3 1. The answer to this query is $3^2 + 7^2 = 58$.

Query 8 after decoding is 2 6 6 2. The answer to this query is $6^2 = 36$.

Sample Input 1	Sample Output 1
7 8 1 2 1 3 2 1 2 2 1 6 1 3 6 5 4 2 3 3 6 2 1 7 5 1 6 2 6 1 1 3 7 6 5 2 4 4 7	46 45 0 0 10 58 36

Problem I
Infinite Fraction Sequence

In this problem, you'll have to answer T queries. For each query, you are given n and k. Consider an infinite sequence of fractions p/q where $1 \leq p, 1 \leq q \leq n$. These fractions are arranged in ascending order, primarily sorted by the value p/q and, in case of a tie, by the numerator p. Your task is to find the k-th smallest fraction in this sequence.

Input

The first line contains an integer T $(1 \leq T \leq 5000)$, the number of queries. Each of the following T lines contains two integers n and k $(1 \leq n \leq 10^9, 1 \leq k \leq 10^{18})$.

Output

For each query, output a single line containing two integers p and q, separated by a space. These integers represent the k-th smallest fraction p/q.

Sample explanation

When $n = 3$, the first three elements of the infinite sequence are: $1/3$, $1/2$ and $2/3$.

Sample Input 1	Sample Output 1
1 3 3	2 3

This page is intentionally left blank.

Problem J
Jumbled Graph

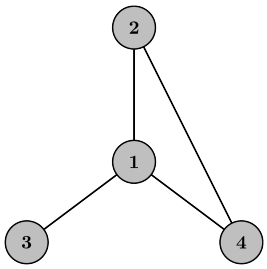
Let's revisit a very basic problem of graph theory: given an connected undirected graph of n vertices (numbered 1 to n), what is the depth-first-search (DFS) order of this graph. The DFS order could be generated by the following pseudocode:

```
dfs_order = []  # empty list

def DFS(u):
    visited[u] = True
    dfs_order.append(u)
    for vertex v that is directly connected to u:
        # note that the order of v is totally random
        if visited[v] == False:
            DFS(v)

DFS(random(1,n))
```

Thus, there are 12 valid DFS orders for the following graph:



1. 1,2,4,3
2. 1,3,2,4
3. 1,3,4,2
4. 1,4,2,3
5. 2,1,3,4
6. 2,1,4,3
7. 2,4,1,3
8. 3,1,2,4
9. 3,1,4,2
10. 4,1,2,3
11. 4,1,3,2
12. 4,2,1,3

Given a permutation of 1 to n, your task is to determine how many undirected graph takes this permutation as a valid DFS order? Note that, two graphs $G1$ and $G2$ are considered different iff there exists two vertices u and v ($u \neq v$) where $G1$ contains an edge between u and v and $G2$ does not contains it; or vice versa.

Input

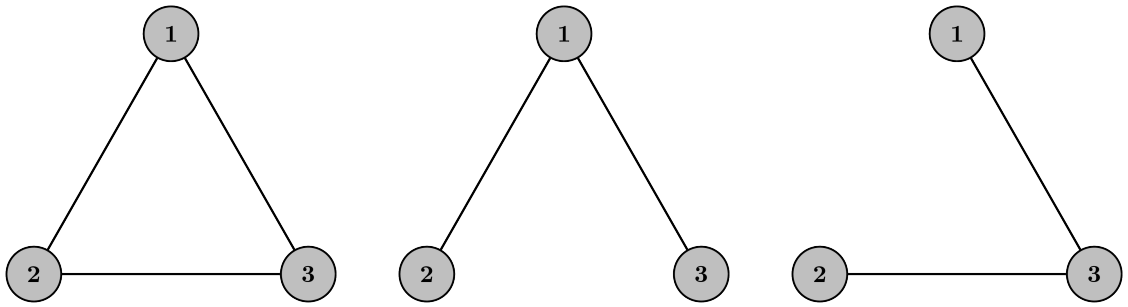
- The first line contains an integer n ($1 \leq n \leq 16$) — the number of vertices.
- The second line contains n integers $a_1, a_2, \ldots a_n$ representing a DFS order. It is guaranteed that $(a_1, a_2, \ldots a_n)$ is a permutation of $(1, 2, \ldots, n)$.

Output

You need to print the number of graphs that takes this permutation to be its DFS order modulo 998 244 353.

Sample explanation

- In the first sample, there is only one graph — the complete graph takes `2, 1` as a valid DFS order.
- In the second sample, these 3 graphs takes `3, 1, 2` as a valid DFS order.



Sample Input 1	Sample Output 1
2 2 1	1

Sample Input 2	Sample Output 2
3 3 1 2	3

Problem K
Knockout Phase

The *UEFA Champions League* is one of the most prestigous football tournaments all over the world. This event is hosted annually, involving top-division European football clubs. The final match of UEFA Champions League usually attracts over 400 million viewers.

According to the current format of the UEFA Champions League, a tournament consists of 3 periods: qualification round, group stage and knockout phase. 32 best European football clubs compete in the group stage, which are divided into 8 groups of four. Winners and runner-ups from these groups advance to the knockout phase. The knockout phase is a single elimination bracket starting with the Round of 16. 16 teams are drawn into 8 matches satisfying the following rules:

- In every match, a group winner plays against a group runner-up.
- Teams from the same group can not be drawn against each other.
- Teams from the same association can not be drawn against each other.

Since several teams can not play against some other teams; for some teams A, B and C, the probability that A play against B may not be equal to the probability that A play against C. As football fans want to predict the opponents of their beloved teams, you are asked to write a program to calculate the probability of a match to occur.

In this problem, we are working on a generalized version of the UEFA Champions League. The group stage consists of n groups instead of 8. Groups are numbered from 1 to n. During the first round of the knockout phase, n group winners and n group runner-ups form n matches. All matches have to satisfy all three above rules. Therefore, a *valid draw* of n matches can be represented by a permutation $x_1, x_2, \ldots, x_n$ of integers from 1 to n, meaning that for every i, the winner from the i-th group plays against the runner-up from the x_i-th group. Each valid draw has the same probability to be chosen.

Given the association of all $2 \cdot n$ teams, your task is to calculate, for every pair of teams, the probability for them to play against the other.

Input

The input consists of multiple test cases. Each test case is presented as below:

- The first line contains an integer n $(1 \leq n \leq 20)$ — the number of groups.
- In the last n lines, the i-th one contains two strings, which denote the associations of the winner and the runner-up from the i-th group, respectively. Each string contains from 1 to 3 uppercase English characters.

The input is terminated by a line containing a single 0 which does not represent a test case.
It is guaranteed that valid draws always exist.
The sum of n over all test cases does not exceed 300.

Output

For each test case, print n lines, each line contains n numbers. The j-th number on the i-th line is the probability that the winner from the i-th group plays against the runner-up from the j-th group.

Your answer will be considered correct if its relative or absolute error doesn't exceed 10^{-6}.

More formally: let's assume that your answer is C, and the answer of the jury is J. The checker program will consider your answer correct, if $\frac{|C-J|}{max(1,J)} \leq 10^{-6}$.

Sample explanation

In the first test case, the winner from the first group can play against the runner-ups from neither the first group (as they are from the same group) nor the second group (as they are from the same association `HN`). Hence, they must play against the runner-up from the third group. Consequently, the winner from the second group must play against the runner-up from the first group, and the winner from the third group must play against the runner-up from the second group. This is the only valid draw.

In the second test case, there are 6 valid draws:

- $(2, 3, 5, 1, 4)$
- $(2, 4, 5, 1, 3)$
- $(2, 5, 4, 1, 3)$
- $(4, 1, 5, 2, 3)$
- $(4, 5, 1, 2, 3)$
- $(4, 5, 2, 1, 3)$

For the winner from the first group, the probability to play against the runner-ups from the second and the forth group are both 0.5.

For the winner from the second group, the probability to play against the runner-ups are $\frac{1}{6}$, 0, $\frac{1}{6}$, $\frac{1}{6}$, $\frac{1}{2}$, respectively.

In the third test case, every group winner can play against every group runner-up (except the one from the same group). Therefore, it can be seen that the probability to play against these runner-ups are equal.

Sample Input 1

```
3
HN LA
QT HN
SG PY
5
LCK LPL
LEC LPL
LCS LCK
LCK VCS
LPL LCK
4
GBR FRA
GER ESP
SUI POR
ITA HUN
0
```

Sample Output 1

```
0.00000000 0.00000000 1.00000000
1.00000000 0.00000000 0.00000000
0.00000000 1.00000000 0.00000000
0.00000000 0.50000000 0.00000000 0.50000000 0.00000000
0.16666667 0.00000000 0.16666667 0.16666667 0.50000000
0.16666667 0.16666667 0.00000000 0.16666667 0.50000000
0.66666667 0.33333333 0.00000000 0.00000000 0.00000000
0.00000000 0.00000000 0.83333333 0.16666667 0.00000000
0.00000000 0.33333333 0.33333333 0.33333333
0.33333333 0.00000000 0.33333333 0.33333333
0.33333333 0.33333333 0.00000000 0.33333333
0.33333333 0.33333333 0.33333333 0.00000000
```

This page is intentionally left blank.

Problem L
Lowest Possible Place

The year is 2050. After several delays due to extraordinary circumstances, the Intercontinental Cooperative Plumbing Competition (ICPC) has finally been held! Participating in the contest are n plumbing teams who qualified via regional plumbing competitions from 2021 all the way to 2049. Traditionally, the ICPC consists of m plumbing problems, where teams compete to solve as many as possible. However, unlike other competitions of the same abbreviation, only the **first** team that solves a problem is given one point for that problem. After all problems have been solved, teams are ranked from 1^{st} to n^{th} by number of points, with teams that have the same point receiving the same ranking. Formally, for each team i, their rank r_i is the number of teams j that scores **strictly more points** than them, plus 1. For example, if there are 5 teams competing on 6 problems and the first teams to solve each problem are 1, 2, 1, 4, 5, and 1, respectively, then the rankings are as follows:

- 1^{st} place: team 1 (3 points)
- 2^{nd} place: teams 2, 4, and 5 (1 point)
- 5^{th} place: team 3 (0 points)

The organizers have prepared m problems. Since each competing team has very particular strengths and weaknesses, the organizers know for certain that team a_i will be the first to solve problem i. Unfortunately, due to extraordinary circumstances (again!), the contest has to be cut short. It was decided that the organizers will select **at most** k **consecutive** problems for the contest. In other words, they will choose two indices f, l such that $1 \le f \le l \le m$ and $1 \le l - f + 1 \le k$; and then select problems $f, f + 1, \ldots, l$ for the contest.

As a member of the organizing team and a former plumber himself, MofK is very eager to know which ranking each team might end up with. While any team can win the tournament, finding the **lowest** (most pestimistic) possible ranking for each team is not a trivial task. Please help him answer this challenging question!

Input

The first line contains three integers n, m, and k ($1 \le n \le 10000$, $1 \le k \le m \le 10000$) — the number of competing teams, the number of prepared problems, and the limit on the number of problems in the contest, respectively.

The second line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$), the first team to solve each problem.

Output

Print n space-separated integers $r_1, r_2, \ldots, r_n$ in one line, where r_i is the lowest possible rank of team i among all possible choices of problems.

Sample explanation

In the example, there are 4 teams and 5 prepared problems, and since $k = m = 5$, it is possible to choose any set of consecutive problems.

- Team 1 will finish last if the organizers choose the problems $[3, 4, 5]$.
- Team 2 will finish third if the organizers choose the problems $[4, 5]$. There is no possible scenario where team 2 finishes last.
- Team 3 will finish third if the organizers choose the problems $[2, 3]$. There is no possible scenario where team 3 finishes last.
- Team 4 will finish last if the organizers choose the problems $[2, 3, 4]$.

Sample Input 1	Sample Output 1
4 5 5 1 1 2 3 4	4 3 3 4

Problem M
MofK's Mysterious Money Making Machine

This is an interactive problem.

MofK has invented an arcade game machine. Inside MofK's machine, there are n secret coins. The states of the coins are represented by a binary string s of length n, where $s_i = 0$ denotes that the i-th coin is tails, and $s_i = 1$ denotes that i-th coin is heads. Since this is a secret coin sequence, no one except MofK knows how the string s looks like.

The player can interact with the machine by inserting coins into it, and if the player can guess the state of the secret coin sequence inside the machine, the machine will reward the player with all the coins inside, including the n secret coins. An interaction with the machine consists of the following steps:

1. First, the player inserts n coins into the machine. The states of these inserted coins are represented by a binary string x, where $x_i = 0$ denotes that the i-th coin is tails, and $x_i = 1$ denotes that the i-th coin is heads.
2. Then, the machine calculates the following values:
   - $cnt_{\texttt{correct}}$ is the number of positions i where $s_i = x_i$.
   - $cnt_{\texttt{incorrect}}$ is the number of positions i where $s_i \neq x_i$.
3. Next, the player presses one of two buttons: the "`?`" button to ask, or the "`!`" button to guess the secret coin sequence:
4. After either of the two buttons is pressed, the machine reacts as below:

If "`?`" button is pressed	The machine will release a **heads** coin if $cnt_{\texttt{correct}} < cnt_{\texttt{incorrect}}$. The machine will release a **tails** coin if $cnt_{\texttt{correct}} > cnt_{\texttt{incorrect}}$.
If "`!`" button is pressed	The machine will release all the coins inside if $cnt_{\texttt{correct}} = n$. The machine will be locked if $cnt_{\texttt{correct}} < n$. The player will not be able to interact with the machine anymore.

You have coins to interact with the machine at most 1024 times, including both asking and guessing interactions. Can you come up with a strategy to retrieve all the coins from MofK's arcade game machine?

Interaction

In this problem, the jury plays the role of MofK's arcade game machine, and your program plays the role of the player of the arcade game machine.

Before the interaction process begins, you need to read an odd integer n ($1 \leq n \leq 999$) – the number of secret coins that MofK has placed into the machine. Then you can interact with the Jury's program as follows:

If you decide to insert n coins into the machine and then press the "`?`" button to ask the machine:

- First, print a single line "`?` x" (without quotation mark), where x is a binary string of length n that describes the state of the coins you will insert into the machine in the above specified format.
- Then, you need to read a number c ($c \in \{0, 1\}$) that describes the state of the coin that the machine releases. $c = 1$ means that the machine releases a **heads** coin, and $c = 0$ means that the machine releases a **tails** coin.

If you decide to insert a coin into the machine and press the "`!`" button to guess the coin sequence:

- First, print a single line "`!` x" (without quotation marks), where x is a binary string of length n that describes the state of the coins you will insert into the machine in the above specified format.
- Then, the interaction process stops immediately. Your program must terminate.

The state sequence of the secret coins s in the machine is fixed before the interaction process begins and will not change during the interaction.

Example Interaction

Standard Input	Standard Output	Explanation
3		There are $n = 3$ secret coins in the machine.
	? 000	You ask the machine with the coin state sequence 000.
1		The machine releases a **heads** coin, which means the number of positions where the hidden string s and 000 match is **less than** the number of positions where they do not match.
	? 001	You ask the machine with the coin state sequence 001.
0		The machine releases a **tails** coin, which means the number of positions where the hidden string s and 001 match is **greater than** the number of positions where they do not match.
	? 010	You ask the machine with the coin state sequence 010.
1		The machine releases a **heads** coin, which means the number of positions where the hidden string s and 010 match is **less than** the number of positions where they do not match.
	? 011	You ask the machine with the coin state sequence 011.
1		The machine releases a **heads** coin, which means the number of positions where the hidden string s and 011 match is **less than** the number of positions where they do not match.
	? 100	You ask the machine with the coin state sequence 100.
0		The machine releases a **tails** coin, which means the number of positions where the hidden string s and 100 match is **greater than** the number of positions where they do not match.
	? 101	You ask the machine with the coin state sequence 101.
0		The machine releases a **tails** coin, which means the number of positions where the hidden string s and 101 match is **greater than** the number of positions where they do not match.
	? 110	You ask the machine with the coin state sequence 110.
1		The machine releases a **heads** coin, which means the number of positions where the hidden string s and 110 match is **less than** the number of positions where they do not match.
	? 111	You ask the machine with the coin state sequence 111.
0		The machine releases a **tails** coin, which means the number of positions where the hidden string s and 111 match is **greater than** the number of positions where they do not match.
	! 101	You guess the secret coin states as *heads up, tails up, heads up* respectively. Your program then terminates.

Note

After printing a line do not forget to output end of line and flush the output. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;
- see documentation for other languages.