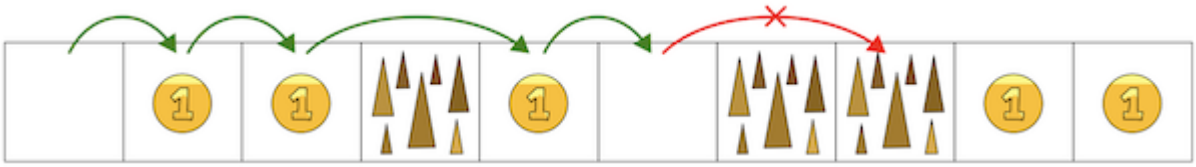
A. Thorns and Coins

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 512 megabytes

During your journey through computer universes, you stumbled upon a very interesting world. It is a path with n consecutive cells, each of which can either be empty, contain thorns, or a coin. In one move, you can move one or two cells along the path, provided that the destination cell does not contain thorns (and belongs to the path). If you move to the cell with a coin, you pick it up.



Here, green arrows correspond to legal moves, and the red arrow corresponds to an illegal move.

You want to collect as many coins as possible. Find the maximum number of coins you can collect in the discovered world if you start in the leftmost cell of the path.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 50$) — the length of the path.

The second line of each test case contains a string of n characters, the description of the path. The character `'.'` denotes an empty cell, `'@'` denotes a cell with a coin, and `'*'` denotes a cell with thorns. It is guaranteed that the first cell is empty.

Output

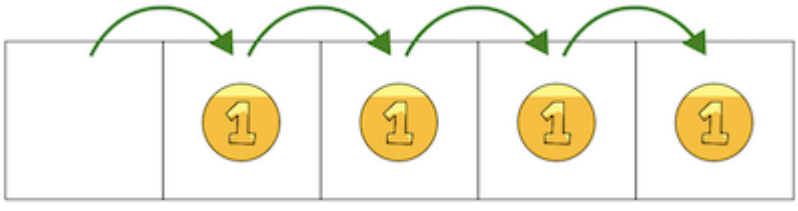
For each test case, output a single integer, the maximum number of coins you can collect.

Standard Input	Standard Output
3 10 .@@*@.**@@ 5 .@@@@ 15 .@@..@***..@@@*	3 4 3

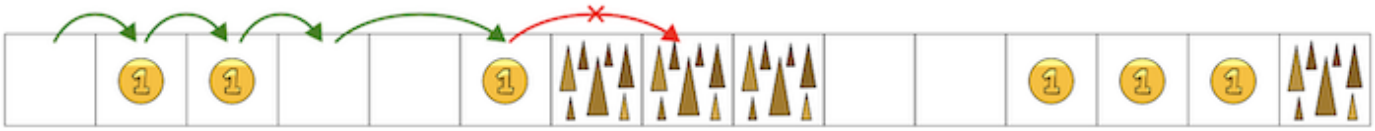
Note

The picture for the first example is in the problem statement.

Here is the picture for the second example:



And here is the picture for the third example:



,

B. Chaya Calendar

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

The Chaya tribe believes that there are n signs of the apocalypse. Over time, it has been found out that the i-th sign occurs every a_i years (in years a_i, $2 \cdot a_i$, $3 \cdot a_i$, $\ldots$).

According to the legends, for the apocalypse to happen, the signs must occur sequentially. That is, first they wait for the first sign to occur, then strictly after it, the second sign will occur, and so on. That is, if the i-th sign occurred in the year x, the tribe starts waiting for the occurrence of the $(i + 1)$-th sign, starting from the year $x + 1$.

In which year will the n-th sign occur and the apocalypse will happen?

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 100$) — the number of signs.

The second line of each test case contains n integers $a_1, a_2, a_3, \ldots, a_n$ ($1 \le a_i \le 10^6$) — the periodicities of the signs.

Output

For each test case, output a single integer — the year in which all n signs will occur.

Standard Input	Standard Output
4 6 3 2 4 5 9 18 5 1 2 3 4 5 5 1 1 1 1 1 6 50 30 711 200 503 1006	36 5 5 2012

Note

In the first set of input data of the example:

- The tribe will wait for the first sign in the 3-rd year;
- the tribe will wait for the second sign in the 4-th year (since year 2 have already passed);
- the tribe will wait for the third sign in the 8-th year (since the second sign has already occurred in the 4-th year);
- the tribe will wait for the fourth sign in the 10-th year (since year 5 have already passed);
- the tribe will wait for the fifth sign in the 18-th year (since year 9 have already passed);
- the tribe will wait for the sixth sign in the 36-th year (since the fifth sign has already occurred in the 18-th year).

,

C. LR-remainders

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

You are given an array a of length n, a positive integer m, and a string of commands of length n. Each command is either the character 'L' or the character 'R'.

Process all n commands in the order they are written in the string s. Processing a command is done as follows:

- First, output the remainder of the product of all elements of the array a when divided by m.
- Then, if the command is 'L', remove the leftmost element from the array a, if the command is 'R', remove the rightmost element from the array a.

Note that after each move, the length of the array a decreases by 1, and after processing all commands, it will be empty.

Write a program that will process all commands in the order they are written in the string s (from left to right).

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. Then descriptions of t test cases follow.

Each test case of the input is given by three lines.

The first line contains two integers n and m ($1 \le n \le 2 \cdot 10^5, 1 \le m \le 10^4$) — the initial length of the array a and the value to take the remainder by.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^4$) — the elements of the array a.

The third line contains a string s consisting of n characters 'L' and 'R'.

It is guaranteed that the sum of the values of n for all test cases in a test does not exceed $2 \cdot 10^5$.

Output

For each test case, output n integers $b_1, b_2, \ldots, b_n$, where b_i is the remainder when dividing the product of all elements of the current state of the array a by m at the beginning of the execution of the i-th command.

Standard Input	Standard Output
4 4 6 3 1 4 2 LRRL 5 1 1 1 1 1 1 LLLLL 6 8 1 2 3 4 5 6 RLLLRR	0 2 4 1 0 0 0 0 0 0 0 0 4 4 4 0

1 10000 10000 R	

Note

In the first test case of the example:

- $3 \cdot 1 \cdot 4 \cdot 2 \bmod 6 = 24 \bmod 6 = 0$;
- $s_1 = \mathrm{L}$, so we remove the first element and get the array $[1, 4, 2]$;
- $1 \cdot 4 \cdot 2 \bmod 6 = 8 \bmod 6 = 2$;
- $s_2 = \mathrm{R}$, so we remove the last element and get the array $[1, 4]$;
- $1 \cdot 4 \bmod 6 = 4 \bmod 6 = 4$;
- $s_3 = \mathrm{R}$, so we remove the last element and get the array $[1]$;
- $1 \bmod 6 = 1$;
- $s_4 = \mathrm{L}$, so we remove the first element and get an empty array.

,

D. Card Game

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Two players are playing an online card game. The game is played using a 32-card deck. Each card has a suit and a rank. There are four suits: clubs, diamonds, hearts, and spades. We will encode them with characters 'C', 'D', 'H', and 'S', respectively. And there are 8 ranks, in increasing order: '2', '3', '4', '5', '6', '7', '8', '9'.

Each card is denoted by two letters: its rank and its suit. For example, the 8 of Hearts is denoted as 8H.

At the beginning of the game, one suit is chosen as the **trump suit**.

In each round, players make moves like this: the first player places one of his cards on the table, and the second player must beat this card with one of their cards. After that, both cards are moved to the discard pile.

A card can beat another card if both cards have the same suit and the first card has a higher rank than the second. For example, 8S can beat 4S. Additionally, a trump card can beat any non-trump card, regardless of the rank of the cards, for example, if the trump suit is clubs ('C'), then 3C can beat 9D. Note that trump cards can be beaten only by the trump cards of higher rank.

There were n rounds played in the game, so the discard pile now contains $2n$ cards. You want to reconstruct the rounds played in the game, but the cards in the discard pile are shuffled. Find any possible sequence of n rounds that might have been played in the game.

Input

The first line contains integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

The first line of a test case contains the integer number n ($1 \le n \le 16$).

The second line of a test case contains one character, the trump suit. It is one of "CDHS".

The third line of a test case contains the description of $2n$ cards. Each card is described by a two-character string, the first character is the rank of the card, which is one of "23456789", and the second one is the suit of the card, which is one of "CDHS". All cards are different.

Output

For each test case print the answer to it:

- Print n lines. In each line, print the description of two cards, in the same format as in the input: the first card that was played by the first player, and then the card that was used by the second player to beat it.
- If there is no solution, print a single line "IMPOSSIBLE".

If there are multiple solutions, print any of them.

Standard Input	Standard Output
8 3 S 3C 9S 4C 6D 3S 7S	3C 4C 6D 9S 3S 7S IMPOSSIBLE

2 C 3S 5D 9S 6H 1 H 6C 5D 1 S 7S 3S 1 H 9S 9H 1 S 9S 9H 1 C 9D 8H 2 C 9C 9S 6H 8C	IMPOSSIBLE 3S 7S 9S 9H 9H 9S IMPOSSIBLE 6H 9C 9S 8C

,

E. Final Countdown

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

You are in a nuclear laboratory that is about to explode and destroy the Earth. You must save the Earth before the final countdown reaches zero.

The countdown consists of n ($1 \le n \le 4 \cdot 10^5$) mechanical indicators, each showing one decimal digit. You noticed that when the countdown changes its state from x to $x - 1$, it doesn't happen in one move. Instead, each change of a single digit takes one second.

So, for example, if the countdown shows `42`, then it will change to `41` in one second, because only one digit is changed, but if the countdown shows `2300`, then it will change to `2299` in three seconds, because the three last digits are changed.

Find out how much time is left before the countdown reaches zero.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 4 \cdot 10^5$).

The second line contains a string of n digits, the current state of the countdown. It is guaranteed that at least one digit is not zero.

The sum of n for all tests does not exceed $4 \cdot 10^5$.

Output

For each test case, print a single integer without leading zeroes, the number of seconds left before the countdown reaches zero. Note that this number may be huge.

Standard Input	Standard Output
5 2 42 5 12345 2 99 4 0005 27 456480697259671309012631002	46 13715 108 5 507200774732968121125145546

Note

In the first example, there are four changes that take 2 seconds: `40` to `39`, `30` to `29`, `20` to `19`, and `10` to `09`, other changes take 1 second each. So the total time is $2 \cdot 4 + 1 \cdot (42 - 4) = 46$.

,

F. Feed Cats

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 512 megabytes

There is a fun game where you need to feed cats that come and go. The level of the game consists of n steps. There are m cats; the cat i is present in steps from l_i to r_i, inclusive. In each step, you can feed all the cats that are currently present or do nothing.

If you feed the same cat more than once, it will overeat, and you will immediately lose the game. Your goal is to feed as many cats as possible without causing any cat to overeat.

Find the maximum number of cats you can feed.

Formally, you need to select several integer points from the segment from 1 to n in such a way that among given segments, none covers two or more of the selected points, and as many segments as possible cover one of the selected points.

Input

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

The first line of each test case contains two integers n and m ($1 \le n \le 10^6$, $1 \le m \le 2 \cdot 10^5$).

The i-th of the next m lines contains a pair of integers l_i and r_i ($1 \le l_i \le r_i \le n$).

The sum of n for all tests does not exceed 10^6, the sum of m for all tests does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer, the maximum number of cats you can feed.

Standard Input	Standard Output
3 15 6 2 10 3 5 2 4 7 7 8 12 11 11 1000 1 1 1000 5 10 1 2 3 4 3 4 3 4 3 4 1 1	5 1 10

1 2 3 3 3 4 3 4	

Note

In the first example, one of the ways to feed five cats is to feed at steps 4 and 11.

- At step 4, cats 1, 2, and 3 will be fed.
- At step 11, cats 5 and 6 will be fed.

,

G. Moving Platforms

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 512 megabytes

There is a game where you need to move through a labyrinth. The labyrinth consists of n platforms, connected by m passages.

Each platform is at some level l_i, an integer number from 0 to $H - 1$. In a single step, if you are currently on platform i, you can stay on it, or move to another platform j. To move to platform j they have to be connected by the passage, and their levels have to be the same, namely $l_i = l_j$.

After each step, the levels of all platforms change. The new level of platform i is calculated as $l'_i = (l_i + s_i) \bmod H$, for all i.

You start on platform 1. Find the minimum number of steps you need to get to platform n.

Input

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

The first line of each test case contains three integers n, m, and H ($2 \le n \le 10^5$, $1 \le m \le 10^5$, $1 \le H \le 10^9$).

The second line contains n integers l_i, the initial level of each platform ($0 \le l_i \le H - 1$).

The third line contains n integers s_i, the change of level for each platform ($0 \le s_i \le H - 1$).

Next m lines contain a description of the passages. Each passage is described as a pair of integers — the platforms, connected by the passage. There is at most one passage connecting each pair of platforms, and there is no passage connecting a platform to itself.

The sum of n for all tests does not exceed 10^5, the sum of m for all tests does not exceed 10^5.

Output

For each test case, print a single integer, the minimum number of steps needed to get from platform 1 to platform n.

If it is impossible to get to platform n, print -1.

Standard Input	Standard Output
3 3 3 10 1 9 4 2 3 0 1 2 3 2 1 3 2 1 10 1 2	6 -1 52

```
4 6
1 2
8 7 25
22 14 5 3 10 14 11 1
9 5 4 10 7 16 18 18
2 8
6 3
3 5
7 5
2 6
1 4
4 7
```

Note

This is how levels of the platforms change, and what actions we need to perform in the first example.

	Platform 1	Platform 2	Platform 3	Action
Step 1	1	9	4	Stay on the platform 1
Step 2	3	2	4	Stay on the platform 1
Step 3	**5**	**5**	4	Move to the platform 2
Step 4	7	8	4	Stay on the platform 2
Step 5	9	1	4	Stay on the platform 2
Step 6	1	**4**	**4**	Move to the platform 3