A. K-divisible Sum

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two integers n and k.

You should create an array of n **positive integers** $a_1, a_2, \dots, a_n$ such that the sum $(a_1 + a_2 + \dots + a_n)$ is divisible by k and maximum element in a is minimum possible.

What is the minimum possible maximum element in a?

Input

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

The first and only line of each test case contains two integers n and k ($1 \le n \le 10^9$; $1 \le k \le 10^9$).

Output

For each test case, print one integer — the minimum possible maximum element in array a such that the sum $(a_1 + \dots + a_n)$ is divisible by k.

Example

input

```
4
1 5
4 3
8 8
8 17
```

output

```
5
2
1
3
```

Note

In the first test case $n = 1$, so the array consists of one element a_1 and if we make $a_1 = 5$ it will be divisible by $k = 5$ and the minimum possible.

In the second test case, we can create array $a = [1, 2, 1, 2]$. The sum is divisible by $k = 3$ and the maximum is equal to 2.

In the third test case, we can create array $a = [1, 1, 1, 1, 1, 1, 1, 1]$. The sum is divisible by $k = 8$ and the maximum is equal to 1.

B. Inflation

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a statistic of price changes for one product represented as an array of n positive integers $p_0, p_1, \dots, p_{n-1}$, where p_0 is the initial price of the product and p_i is how the price was increased during the i-th month.

Using these price changes you are asked to calculate the inflation coefficients for each month as the ratio of current price increase p_i to the price at the start of this month $(p_0 + p_1 + \dots + p_{i-1})$.

Your boss said you clearly that the inflation coefficients must not exceed k %, so you decided to **increase** some values p_i in such a way, that all p_i remain integers and the inflation coefficients for each month don't exceed k %.

You know, that the bigger changes — the more obvious cheating. That's why you need to minimize the total sum of changes.

What's the minimum total sum of changes you need to make all inflation coefficients not more than k %?

Input

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

The first line of each test case contains two integers n and k ($2 \le n \le 100$; $1 \le k \le 100$) — the length of array p and coefficient k.

The second line of each test case contains n integers $p_0, p_1, \dots, p_{n-1}$ ($1 \le p_i \le 10^9$) — the array p.

Output

For each test case, print the minimum total sum of changes you need to make all inflation coefficients not more than k %.

Example

input

```
2
4 1
20100 1 202 202
3 100
1 1 1
```

output

```
99
0
```

Note

In the first test case, you can, for example, increase p_0 by 50 and p_1 by 49 and get array $[20150, 50, 202, 202]$. Then you get the next inflation coefficients:

1. $\frac{50}{20150} \le \frac{1}{100}$;
2. $\frac{202}{20150+50} \le \frac{1}{100}$;
3. $\frac{202}{20200+202} \le \frac{1}{100}$;

In the second test case, you don't need to modify array p, since the inflation coefficients are already good:

1. $\frac{1}{1} \le \frac{100}{100}$;
2. $\frac{1}{1+1} \le \frac{100}{100}$;

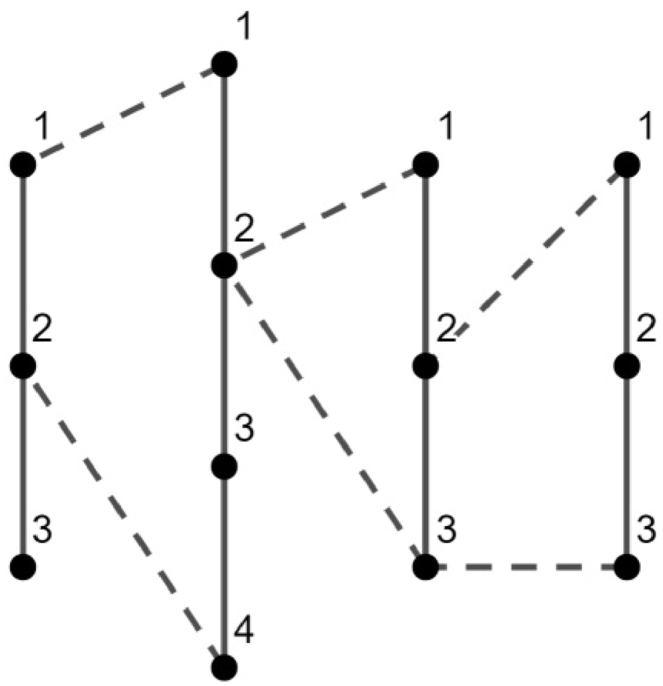
C. Longest Simple Cycle

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have n chains, the i-th chain consists of c_i vertices. Vertices in each chain are numbered independently from 1 to c_i along the chain. In other words, the i-th chain is the undirected graph with c_i vertices and $(c_i - 1)$ edges connecting the j-th and the $(j+1)$-th vertices for each $1 \le j < c_i$.

Now you decided to unite chains in one graph in the following way:

1. the first chain is skipped;
2. the 1-st vertex of the i-th chain is connected by an edge with the a_i-th vertex of the $(i-1)$-th chain;
3. the last (c_i-th) vertex of the i-th chain is connected by an edge with the b_i-th vertex of the $(i-1)$-th chain.



Picture of the first test case. Dotted lines are the edges added during uniting process

Calculate the length of the longest simple cycle in the resulting graph.

A *simple cycle* is a chain where the first and last vertices are connected as well. If you travel along the simple cycle, each vertex of this cycle will be visited exactly once.

Input

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

The first line of each test case contains the single integer n ($2 \le n \le 10^5$) — the number of chains you have.

The second line of each test case contains n integers $c_1, c_2, \dots, c_n$ ($2 \le c_i \le 10^9$) — the number of vertices in the corresponding chains.

The third line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($a_1 = -1$; $1 \le a_i \le c_{i-1}$).

The fourth line of each test case contains n integers $b_1, b_2, \dots, b_n$ ($b_1 = -1$; $1 \le b_i \le c_{i-1}$).

Both a_1 and b_1 are equal to -1, they aren't used in graph building and given just for index consistency. It's guaranteed that the sum of n over all test cases doesn't exceed 10^5.

Output

For each test case, print the length of the longest simple cycle.

Example

input

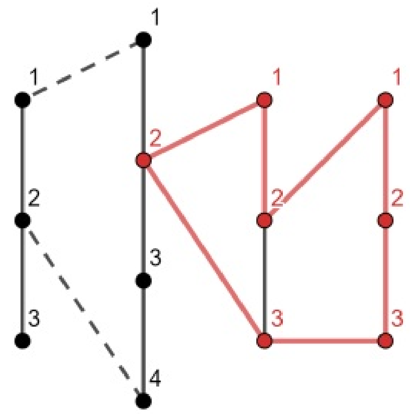
```
3
4
3 4 3 3
-1 1 2 2
-1 2 2 3
2
5 6
-1 5
-1 1
3
3 5 2
-1 1 1
-1 3 5
```

output

```
7
11
8
```

Note

In the first test case, the longest simple cycle is shown below:



We can't increase it with the first chain, since in such case it won't be simple — the vertex 2 on the second chain will break simplicity.

D. Journey

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

There are $n+1$ cities, numbered from 0 to n. n roads connect these cities, the i-th road connects cities $i-1$ and i ($i \in [1, n]$).

Each road has a direction. The directions are given by a string of n characters such that each character is either L or R. If the i-th character is L, it means that the i-th road initially goes from the city i to the city $i-1$; otherwise it goes from the city $i-1$ to the city i.

A traveler would like to visit as many cities of this country as possible. Initially, they will choose some city to start their journey from. Each day, the traveler **must** go from the city where they currently are to a neighboring city using one of the roads, and they can go along a road only if it is directed in the same direction they are going; i. e., if a road is directed from city i to the city $i+1$, it is possible to travel from i to $i+1$, but not from $i+1$ to i. After the traveler moves to a neighboring city, **all** roads change their directions **to the opposite ones**. If the traveler cannot go from their current city to a neighboring city, their journey ends; it is also possible to end the journey whenever the traveler wants to.

The goal of the traveler is to visit as many different cities as possible (they can visit a city multiple times, but only the first visit is counted). For each city i, calculate the maximum number of different cities the traveler can visit during **exactly one journey** if they start in the city i.

Input

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

Each test case consists of two lines. The first line contains one integer n ($1 \le n \le 3 \cdot 10^5$). The second line contains the string s consisting of exactly n characters, each character is either L or R.

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

Output

For each test case, print $n+1$ integers. The i-th integer should be equal to the maximum number of different cities the traveler can visit during one journey if this journey starts in the i-th city.

Example

input

```
2
6
LRRRLL
3
LRL
```

output

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

E. Pattern Matching

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given n patterns $p_1, p_2, \dots, p_n$ and m strings $s_1, s_2, \dots, s_m$. Each pattern p_i consists of k characters that are either lowercase Latin letters or wildcard characters (denoted by underscores). All patterns are pairwise distinct. Each string s_j consists of k lowercase Latin letters.

A string a matches a pattern b if for each i from 1 to k either b_i is a wildcard character or $b_i = a_i$.

You are asked to rearrange the patterns in such a way that the first pattern the j-th string matches is $p[mt_j]$. You are allowed to leave the order of the patterns unchanged.

Can you perform such a rearrangement? If you can, then print any valid order.

Input

The first line contains three integers n, m and k ($1 \le n, m \le 10^5$, $1 \le k \le 4$) — the number of patterns, the number of strings and the length of each pattern and string.

Each of the next n lines contains a pattern — k characters that are either lowercase Latin letters or underscores. All patterns are pairwise distinct.

Each of the next m lines contains a string — k lowercase Latin letters, and an integer mt ($1 \le mt \le n$) — the index of the first pattern the corresponding string should match.

Output

Print "NO" if there is no way to rearrange the patterns in such a way that the first pattern that the j-th string matches is $p[mt_j]$.

Otherwise, print "YES" in the first line. The second line should contain n distinct integers from 1 to n — the order of the patterns. If there are multiple answers, print any of them.

Examples

input

```
5 3 4
_b_d
__b_
aaaa
ab__
_bcd
abcd 4
abba 2
dbcd 5
```

output
YES 3 2 4 5 1

input
1 1 3 __c cba 1

output
NO

input
2 2 2 a_ _b ab 1 ab 2

output
NO

Note

The order of patterns after the rearrangement in the first example is the following:

- aaaa
- __b_
- ab__
- _bcd
- _b_d

Thus, the first string matches patterns ab__, _bcd, _b_d in that order, the first of them is ab__, that is indeed $p[4]$. The second string matches __b_ and ab__, the first of them is __b_, that is $p[2]$. The last string matches _bcd and _b_d, the first of them is _bcd, that is $p[5]$.

The answer to that test is not unique, other valid orders also exist.

In the second example cba doesn't match __c, thus, no valid order exists.

In the third example the order (a_, _b) makes both strings match pattern 1 first and the order (_b, a_) makes both strings match pattern 2 first. Thus, there is no order that produces the result 1 and 2.

F. Lanterns

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n lanterns in a row. The lantern i is placed in position i and has power equal to p_i.

Each lantern can be directed to illuminate either some lanterns to the left or some lanterns to the right. If the i-th lantern is turned to the left, it illuminates all such lanterns j that $j \in [i - p_i, i - 1]$. Similarly, if it is turned to the right, it illuminates all such lanterns j that $j \in [i + 1, i + p_i]$.

Your goal is to choose a direction for each lantern so each lantern is illuminated by at least one other lantern, or report that it is impossible.

Input

The first line contains one integer t ($1 \le t \le 10000$) — the number of test cases.

Each test case consists of two lines. The first line contains one integer n ($2 \le n \le 3 \cdot 10^5$) — the number of lanterns.

The second line contains n integers $p_1, p_2, \dots, p_n$ ($0 \le p_i \le n$) — the power of the i-th lantern.

The sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print the answer as follows:

If it is possible to direct all lanterns so that each lantern is illuminated, print YES in the first line and a string of n characters L and/or R (the i-th character is L if the i-th lantern is turned to the left, otherwise this character is R) in the second line. If there are multiple answers, you may print any of them.

If there is no answer, simply print NO for that test case.

Example

input

```
4
8
0 0 3 1 1 1 1 2
2
1 1
2
2 2
2
0 1
```

output

```
YES
RRLLLLRL
YES
RL
YES
RL
NO
```

G. Minimum Difference

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an integer array a of size n.

You have to perform m queries. Each query has one of two types:

- "$1\ l\ r\ k$" — calculate the minimum value dif such that there are exist k **distinct** integers $x_1, x_2, \dots, x_k$ such that $cnt_i > 0$ (for every $i \in [1, k]$) and $|cnt_i - cnt_j| \le dif$ (for every $i \in [1, k], j \in [1, k]$), where cnt_i is the number of occurrences of x_i in the subarray $a[l..r]$. If it is impossible to choose k integers, report it;
- "$2\ p\ x$" — assign $a_p := x$.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^5$) — the size of the array a and the number of queries.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^5$).

Next m lines contain queries (one per line). Each query has one of two types:

- "$1\ l\ r\ k$" ($1 \le l \le r \le n; 1 \le k \le 10^5$)
- "$2\ p\ x$" ($1 \le p \le n; 1 \le x \le 10^5$).

It's guaranteed that there is at least one query of the first type.

Output

For each query of the first type, print the minimum value of dif that satisfies all required conditions, or -1 if it is impossible to choose k distinct integers.

Example

input

```
12 11
2 1 1 2 1 1 3 2 1 1 3 3
1 2 10 3
1 2 11 3
2 7 2
1 3 9 2
1 1 12 1
1 1 12 4
2 12 4
1 1 12 4
2 1 5
1 3 12 2
1 1 4 3
```

output

```
5
4
1
0
-1
5
0
1
```

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform