Educational Codeforces Round 91 (Rated for Div. 2)

A. Three Indices

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

You are given a permutation $p_1, p_2, \ldots, p_n$. Recall that sequence of n integers is called a *permutation* if it contains all integers from 1 to n exactly once.

Find three indices i, j and k such that:

- $1 \le i < j < k \le n$;
- $p_i < p_j$ and $p_j > p_k$.

Or say that there are no such indices.

Input

The first line contains a single integer T ($1 \le T \le 200$) — the number of test cases.

Next $2T$ lines contain test cases — two lines per test case. The first line of each test case contains the single integer n ($3 \le n \le 1000$) — the length of the permutation p.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$; $p_i \ne p_j$ if $i \ne j$) — the permutation p.

Output

For each test case:

- if there are such indices i, j and k, print YES (case insensitive) and the indices themselves;
- if there are no such indices, print NO (case insensitive).

If there are multiple valid triples of indices, print any of them.

Example

input
3 4 2 1 4 3 6 4 6 1 2 5 3 5 5 3 1 2 4
output
YES 2 3 4 YES 3 5 6 NO

B. Universal Solution

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

Recently, you found a bot to play "Rock paper scissors" with. Unfortunately, the bot uses quite a simple algorithm to play: he has a string $s = s_1 s_2 \ldots s_n$ of length n where each letter is either R, S or P.

While initializing, the bot is choosing a starting index pos ($1 \le pos \le n$), and then it can play any number of rounds. In the first round, he chooses "Rock", "Scissors" or "Paper" based on the value of s_{pos}:

- if s_{pos} is equal to R the bot chooses "Rock";
- if s_{pos} is equal to S the bot chooses "Scissors";
- if s_{pos} is equal to P the bot chooses "Paper";

In the second round, the bot's choice is based on the value of s_{pos+1}. In the third round — on s_{pos+2} and so on. After s_n the bot returns to s_1 and continues his game.

You plan to play n rounds and you've already figured out the string s but still don't know what is the starting index pos. But since

the bot's tactic is so boring, you've decided to find n choices to each round to maximize the average number of wins.

In other words, let's suggest your choices are $c_1c_2 \dots c_n$ and if the bot starts from index pos then you'll win in $win(pos)$ rounds. Find $c_1c_2 \dots c_n$ such that $\frac{win(1)+win(2)+\dots+win(n)}{n}$ is maximum possible.

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

Next t lines contain test cases — one per line. The first and only line of each test case contains string $s = s_1s_2 \dots s_n$ ($1 \le n \le 2 \cdot 10^5$; $s_i \in \{\text{R}, \text{S}, \text{P}\}$) — the string of the bot.

It's guaranteed that the total length of all strings in one test doesn't exceed $2 \cdot 10^5$.

Output
For each test case, print n choices $c_1c_2 \dots c_n$ to maximize the average number of wins. Print them in the same manner as the string s.

If there are multiple optimal answers, print any of them.

Example

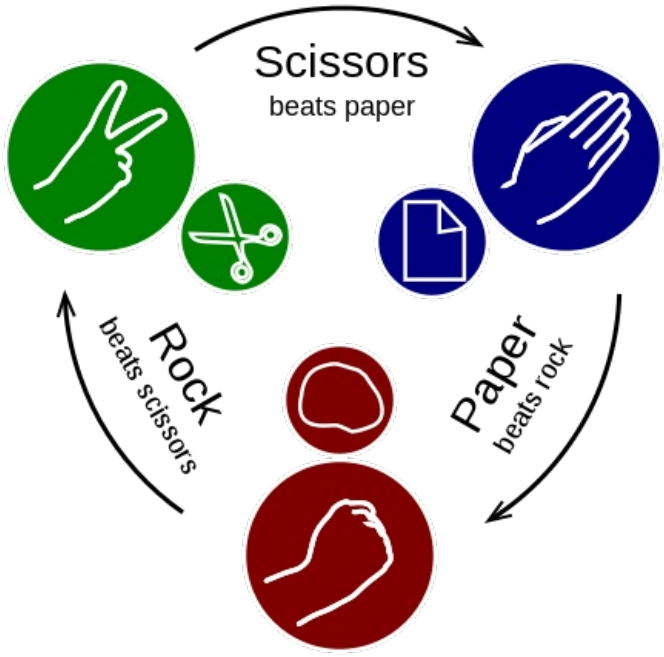
input
3 RRRR RSP S
output
PPPP RSP R

Note
In the first test case, the bot (wherever it starts) will always choose "Rock", so we can always choose "Paper". So, in any case, we will win all $n = 4$ rounds, so the average is also equal to 4.

In the second test case:

- if bot will start from $pos = 1$, then (s_1, c_1) is draw, (s_2, c_2) is draw and (s_3, c_3) is draw, so $win(1) = 0$;
- if bot will start from $pos = 2$, then (s_2, c_1) is win, (s_3, c_2) is win and (s_1, c_3) is win, so $win(2) = 3$;
- if bot will start from $pos = 3$, then (s_3, c_1) is lose, (s_1, c_2) is lose and (s_2, c_3) is lose, so $win(3) = 0$;

The average is equal to $\frac{0+3+0}{3} = 1$ and it can be proven that it's the maximum possible average.
A picture from Wikipedia explaining "Rock paper scissors" game:



C. Create The Teams

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

There are n programmers that you want to split into several non-empty teams. The skill of the i-th programmer is a_i. You want to

assemble the maximum number of teams from them. There is a restriction for each team: the number of programmers in the team multiplied by the minimum skill among all programmers in the team must be at least x.

Each programmer should belong to at most one team. Some programmers may be left without a team.

Calculate the maximum number of teams that you can assemble.

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

The first line of each test case contains two integers n and x ($1 \le n \le 10^5; 1 \le x \le 10^9$) — the number of programmers and the restriction of team skill respectively.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the skill of the i-th programmer.

The sum of n over all inputs does not exceed 10^5.

Output
For each test case print one integer — the maximum number of teams that you can assemble.

Example

input
3 5 10 7 11 2 9 5 4 8 2 4 2 3 4 11 1 3 3 7
output
2 1 0

D. Berserk And Fireball

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

There are n warriors in a row. The power of the i-th warrior is a_i. All powers are pairwise distinct.

You have two types of spells which you may cast:

1. Fireball: you spend x mana and destroy **exactly** k consecutive warriors;
2. Berserk: you spend y mana, choose two consecutive warriors, and the warrior with greater power destroys the warrior with smaller power.

For example, let the powers of warriors be $[2, 3, 7, 8, 11, 5, 4]$, and $k = 3$. If you cast Berserk on warriors with powers 8 and 11, the resulting sequence of powers becomes $[2, 3, 7, 11, 5, 4]$. Then, for example, if you cast Fireball on consecutive warriors with powers $[7, 11, 5]$, the resulting sequence of powers becomes $[2, 3, 4]$.

You want to turn the current sequence of warriors powers $a_1, a_2, \ldots, a_n$ into $b_1, b_2, \ldots, b_m$. Calculate the minimum amount of mana you need to spend on it.

Input
The first line contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the length of sequence a and the length of sequence b respectively.

The second line contains three integers x, k, y ($1 \le x, y, \le 10^9; 1 \le k \le n$) — the cost of fireball, the range of fireball and the cost of berserk respectively.

The third line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$). It is guaranteed that all integers a_i are pairwise distinct.

The fourth line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le n$). It is guaranteed that all integers b_i are pairwise distinct.

Output
Print the minimum amount of mana for turning the sequnce $a_1, a_2, \ldots, a_n$ into $b_1, b_2, \ldots, b_m$, or -1 if it is impossible.

Examples

input
5 2 5 2 3 3 1 4 5 2 3 5

output
8

input
4 4 5 1 4 4 3 1 2 2 4 3 1
output
-1

input
4 4 2 1 11 1 3 2 4 1 3 2 4
output
0

E. Merging Towers

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

You have a set of n discs, the i-th disc has radius i. Initially, these discs are split among m towers: each tower contains at least one disc, and the discs in each tower are sorted in descending order of their radii from bottom to top.

You would like to assemble one tower containing all of those discs. To do so, you may choose two different towers i and j (each containing at least one disc), take several (possibly all) top discs from the tower i and put them on top of the tower j in the same order, as long as the top disc of tower j is bigger than each of the discs you move. You may perform this operation any number of times.

For example, if you have two towers containing discs $[6, 4, 2, 1]$ and $[8, 7, 5, 3]$ (in order from bottom to top), there are only two possible operations:

- move disc 1 from the first tower to the second tower, so the towers are $[6, 4, 2]$ and $[8, 7, 5, 3, 1]$;
- move discs $[2, 1]$ from the first tower to the second tower, so the towers are $[6, 4]$ and $[8, 7, 5, 3, 2, 1]$.

Let the *difficulty* of some set of towers be the minimum number of operations required to assemble one tower containing all of the discs. For example, the *difficulty* of the set of towers $[[3, 1], [2]]$ is 2: you may move the disc 1 to the second tower, and then move both discs from the second tower to the first tower.

You are given $m - 1$ queries. Each query is denoted by two numbers a_i and b_i, and means "merge the towers a_i and b_i" (that is, take all discs from these two towers and assemble a new tower containing all of them in descending order of their radii from top to bottom). The resulting tower gets index a_i.

For each $k \in [0, m - 1]$, calculate the *difficulty* of the set of towers after the first k queries are performed.

Input

The first line of the input contains two integers n and m ($2 \le m \le n \le 2 \cdot 10^5$) — the number of discs and the number of towers, respectively.

The second line contains n integers t_1, t_2, ..., t_n ($1 \le t_i \le m$), where t_i is the index of the tower disc i belongs to. Each value from 1 to m appears in this sequence at least once.

Then $m - 1$ lines follow, denoting the queries. Each query is represented by two integers a_i and b_i ($1 \le a_i, b_i \le m$, $a_i \ne b_i$), meaning that, during the i-th query, the towers with indices a_i and b_i are merged (a_i and b_i are chosen in such a way that these towers exist before the i-th query).

Output

Print m integers. The k-th integer (0-indexed) should be equal to the difficulty of the set of towers after the first k queries are performed.

Example

input
7 4 1 2 3 3 1 4 3 3 1 2 3 2 4
output

```
5
4
2
0
```

Note

The towers in the example are:

- before the queries: $[[5, 1], [2], [7, 4, 3], [6]]$;
- after the first query: $[[2], [7, 5, 4, 3, 1], [6]]$;
- after the second query: $[[7, 5, 4, 3, 2, 1], [6]]$;
- after the third query, there is only one tower: $[7, 6, 5, 4, 3, 2, 1]$.

F. Strange Addition

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

Let a and b be some non-negative integers. Let's define *strange addition* of a and b as following:

1. write down the numbers one under another and align them by their least significant digit;
2. add them up digit by digit and concatenate the respective sums together.

Assume that both numbers have an infinite number of leading zeros.

For example, let's take a look at a *strange addition* of numbers 3248 and 908:

3	2	4	8
	9	0	8
3	1	1	4 1 6

You are given a string c, consisting of n digits from 0 to 9. You are also given m updates of form:

- x d — replace the digit at the x-th position of c with a digit d.

Note that string c might have leading zeros at any point of time.

After each update print the number of pairs (a, b) such that both a and b are non-negative integers and the result of a *strange addition* of a and b is equal to c.

Note that the numbers of pairs can be quite large, so print them modulo 998244353.

Input

The first line contains two integers n and m ($1 \le n, m \le 5 \cdot 10^5$) — the length of the number c and the number of updates.

The second line contains a string c, consisting of exactly n digits from 0 to 9.

Each of the next m lines contains two integers x and d ($1 \le x \le n$, $0 \le d \le 9$) — the descriptions of updates.

Output

Print m integers — the i-th value should be equal to the number of pairs (a, b) such that both a and b are non-negative integers and the result of a *strange addition* of a and b is equal to c after i updates are applied.

Note that the numbers of pairs can be quite large, so print them modulo 998244353.

Example

input
2 3 14 2 4 2 1 1 0
output
15 12 2

Note

After the first update c is equal to 14. The pairs that sum up to 14 are: $(0, 14)$, $(1, 13)$, $(2, 12)$, $(3, 11)$, $(4, 10)$, $(5, 9)$, $(6, 8)$, $(7, 7)$, $(8, 6)$, $(9, 5)$, $(10, 4)$, $(11, 3)$, $(12, 2)$, $(13, 1)$, $(14, 0)$.

After the second update c is equal to 11.

After the third update c is equal to 01.

G. Circular Dungeon

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

You are creating a level for a video game. The level consists of n rooms placed in a circle. The rooms are numbered 1 through n. Each room contains exactly one exit: completing the j-th room allows you to go the $(j+1)$-th room (and completing the n-th room allows you to go the 1-st room).

You are given the description of the multiset of n chests: the i-th chest has treasure value c_i.

Each chest can be of one of two types:

- regular chest — when a player enters a room with this chest, he grabs the treasure and proceeds to the next room;
- mimic chest — when a player enters a room with this chest, the chest eats him alive, and he loses.

The player starts in a random room with each room having an equal probability of being chosen. The players earnings is equal to the total value of treasure chests he'd collected before he lost.

You are allowed to choose the order the chests go into the rooms. For each k from 1 to n place the chests into the rooms in such a way that:

- each room contains **exactly** one chest;
- **exactly** k chests are mimics;
- the expected value of players earnings is **minimum** possible.

Please note that for each k the placement is chosen independently.

It can be shown that it is in the form of $\frac{P}{Q}$ where P and Q are non-negative integers and $Q \neq 0$. Report the values of $P \cdot Q^{-1} \pmod{998244353}$.

Input

The first contains a single integer n ($2 \le n \le 3 \cdot 10^5$) — the number of rooms and the number of chests.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^6$) — the treasure values of each chest.

Output

Print n integers — the k -th value should be equal to the minimum possible expected value of players earnings if the chests are placed into the rooms in some order and exactly k of the chests are mimics.

It can be shown that it is in the form of $\frac{P}{Q}$ where P and Q are non-negative integers and $Q \neq 0$. Report the values of $P \cdot Q^{-1} \pmod{998244353}$.

Examples

input
2 1 2
output
499122177 0

input
8 10 4 3 6 5 10 7 5
output
499122193 249561095 249561092 873463811 499122178 124780545 623902721 0

Note

In the first example the exact values of minimum expected values are: $\frac{1}{2}$, $\frac{0}{2}$.

In the second example the exact values of minimum expected values are: $\frac{132}{8}$, $\frac{54}{8}$, $\frac{30}{8}$, $\frac{17}{8}$, $\frac{12}{8}$, $\frac{7}{8}$, $\frac{3}{8}$, $\frac{0}{8}$.