Harbour.Space Scholarship Contest 2021-2022 (open for everyone, rated, Div. 1 + Div. 2)

A. Digits Sum

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

Let's define $S(x)$ to be the sum of digits of number x written in decimal system. For example, $S(5) = 5$, $S(10) = 1$, $S(322) = 7$.

We will call an integer x **interesting** if $S(x + 1) < S(x)$. In each test you will be given one integer n. Your task is to calculate the number of integers x such that $1 \le x \le n$ and x is interesting.

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

Then t lines follow, the i-th line contains one integer n ($1 \le n \le 10^9$) for the i-th test case.

Output
Print t integers, the i-th should be the answer for the i-th test case.

Example

input
5 1 9 10 34 880055535
output
0 1 1 3 88005553

Note
The first interesting number is equal to 9.

B. Reverse String

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a string s and a chip, which you can place onto any character of this string.

After placing the chip, you move it to the right several (maybe zero) times, i. e. you perform the following operation several times: if the current position of the chip is i, you move it to the position $i + 1$. Of course, moving the chip to the right is impossible if it is already in the last position.

After moving the chip to the right, you move it to the left several (maybe zero) times, i. e. you perform the following operation several times: if the current position of the chip is i, you move it to the position $i - 1$. Of course, moving the chip to the left is impossible if it is already in the first position.

When you place a chip or move it, you write down the character where the chip ends up after your action. For example, if s is abcdef, you place the chip onto the 3-rd character, move it to the right 2 times and then move it to the left 3 times, you write down the string cdedcb.

You are given two strings s and t. Your task is to determine whether it's possible to perform the described operations with s so that you write down the string t as a result.

Input
The first line contains one integer q ($1 \le q \le 500$) — the number of test cases.

Each test case consists of two lines. The first line contains the string s ($1 \le |s| \le 500$), the second line contains the string t ($1 \le |t| \le 2 \cdot |s| - 1$). Both strings consist of lowercase English characters.

It is guaranteed that the sum of $|s|$ over all test cases does not exceed 500.

Output
For each test case, print "YES" if you can obtain the string t by performing the process mentioned in the statement with the string s, or "NO" if you cannot.

You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Example

input
6 abcdef cdedcb aaa aaaaa aab baaa ab b abcdef abcdef ba baa
output
YES YES NO YES YES NO

Note
Consider the examples.

The first test case is described in the statement.

In the second test case, you can place the chip on the 1-st position, move it twice to the right, and then move it twice to the left.

In the fourth test case, you can place the chip on the 2-nd position, and then don't move it at all.

In the fifth test case, you can place the chip on the 1-st position, move it 5 times to the right, and then finish the process.

C. Penalty

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Consider a simplified penalty phase at the end of a football match.

A penalty phase consists of at most 10 kicks, the first team takes the first kick, the second team takes the second kick, then the first team takes the third kick, and so on. The team that scores more goals wins; if both teams score the same number of goals, the game results in a tie (**note that it goes against the usual football rules**). The penalty phase is stopped if one team has scored more goals than the other team could reach with all of its remaining kicks. For example, if after the 7-th kick the first team has scored 1 goal, and the second team has scored 3 goals, the penalty phase ends — the first team cannot reach 3 goals.

You know which player will be taking each kick, so you have your predictions for each of the 10 kicks. These predictions are represented by a string s consisting of 10 characters. Each character can either be `1`, `0`, or `?`. This string represents your predictions in the following way:

- if s_i is `1`, then the i-th kick will definitely score a goal;
- if s_i is `0`, then the i-th kick definitely won't score a goal;
- if s_i is `?`, then the i-th kick could go either way.

Based on your predictions, you have to calculate the minimum possible number of kicks there can be in the penalty phase (that means, the earliest moment when the penalty phase is stopped, considering all possible ways it could go). Note that **the referee doesn't take into account any predictions when deciding to stop the penalty phase** — you may know that some kick will/won't be scored, but the referee doesn't.

Input
The first line contains one integer t ($1 \le t \le 1\,000$) — the number of test cases.

Each test case is represented by one line containing the string s, consisting of exactly 10 characters. Each character is either `1`, `0`, or `?`.

Output
For each test case, print one integer — the minimum possible number of kicks in the penalty phase.

Example

input
4 1?0???1001 1111111111 ?????????? 0100000000
output
7 10 6 9

Note

Consider the example test:

In the first test case, consider the situation when the 1-st, 5-th and 7-th kicks score goals, and kicks 2, 3, 4 and 6 are unsuccessful. Then the current number of goals for the first team is 3, for the second team is 0, and the referee sees that the second team can score at most 2 goals in the remaining kicks. So the penalty phase can be stopped after the 7-th kick.

In the second test case, the penalty phase won't be stopped until all 10 kicks are finished.

In the third test case, if the first team doesn't score any of its three first kicks and the second team scores all of its three first kicks, then after the 6-th kick, the first team has scored 0 goals and the second team has scored 3 goals, and the referee sees that the first team can score at most 2 goals in the remaining kicks. So, the penalty phase can be stopped after the 6-th kick.

In the fourth test case, even though you can predict the whole penalty phase, the referee understands that the phase should be ended only after the 9-th kick.

D. Backspace

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

You are given two strings s and t, both consisting of lowercase English letters. You are going to type the string s character by character, from the first character to the last one.

When typing a character, instead of pressing the button corresponding to it, you can press the "Backspace" button. It deletes the last character you have typed among those that aren't deleted yet (or does nothing if there are no characters in the current string). For example, if s is "abcbd" and you press Backspace instead of typing the first and the fourth characters, you will get the string "bd" (the first press of Backspace deletes no character, and the second press deletes the character 'c'). Another example, if s is "abcaa" and you press Backspace instead of the last two letters, then the resulting text is "a".

Your task is to determine whether you can obtain the string t, if you type the string s and press "Backspace" instead of typing several (maybe zero) characters of s.

Input

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

The first line of each test case contains the string s ($1 \le |s| \le 10^5$). Each character of s is a lowercase English letter.

The second line of each test case contains the string t ($1 \le |t| \le 10^5$). Each character of t is a lowercase English letter.

It is guaranteed that the total number of characters in the strings over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "YES" if you can obtain the string t by typing the string s and replacing some characters with presses of "Backspace" button, or "NO" if you cannot.

You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Example

input
4 ababa ba ababa bb aaa aaaa aababa ababa
output
YES NO

```
NO
YES
```

Note

Consider the example test from the statement.

In order to obtain "ba" from "ababa", you may press Backspace instead of typing the first and the fourth characters.

There's no way to obtain "bb" while typing "ababa".

There's no way to obtain "aaaa" while typing "aaa".

In order to obtain "ababa" while typing "aababa", you have to press Backspace instead of typing the first character, then type all the remaining characters.

E. Permutation Shift

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

An identity permutation of length n is an array $[1, 2, 3, \ldots, n]$.

We performed the following operations to an identity permutation of length n:

- firstly, we cyclically shifted it to the right by k positions, where k is unknown to you (the only thing you know is that $0 \le k \le n - 1$). When an array is cyclically shifted to the right by k positions, the resulting array is formed by taking k last elements of the original array (without changing their relative order), and then appending $n - k$ first elements to the right of them (without changing relative order of the first $n - k$ elements as well). For example, if we cyclically shift the identity permutation of length 6 by 2 positions, we get the array $[5, 6, 1, 2, 3, 4]$;
- secondly, we performed the following operation **at most** m times: pick any two elements of the array and swap them.

You are given the values of n and m, and the resulting array. Your task is to find all possible values of k in the cyclic shift operation.

Input

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

Each test case consists of two lines. The first line contains two integers n and m ($3 \le n \le 3 \cdot 10^5$; $0 \le m \le \frac{n}{3}$).

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$, each integer from 1 to n appears in this sequence exactly once) — the resulting array.

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

Output

For each test case, print the answer in the following way:

- firstly, print one integer r ($0 \le r \le n$) — the number of possible values of k for the cyclic shift operation;
- secondly, print r integers $k_1, k_2, \ldots, k_r$ ($0 \le k_i \le n - 1$) — all possible values of k **in increasing order**.

Example

input
4 4 1 2 3 1 4 3 1 1 2 3 3 1 3 2 1 6 0 1 2 3 4 6 5
output
1 3 1 0 3 0 1 2 0

Note

Consider the example:

- in the first test case, the only possible value for the cyclic shift is 3. If we shift $[1, 2, 3, 4]$ by 3 positions, we get $[2, 3, 4, 1]$. Then we can swap the 3-rd and the 4-th elements to get the array $[2, 3, 1, 4]$;
- in the second test case, the only possible value for the cyclic shift is 0. If we shift $[1, 2, 3]$ by 0 positions, we get $[1, 2, 3]$. Then we don't change the array at all (we stated that we made **at most** 1 swap), so the resulting array stays $[1, 2, 3]$;
- in the third test case, all values from 0 to 2 are possible for the cyclic shift:

- if we shift $[1, 2, 3]$ by 0 positions, we get $[1, 2, 3]$. Then we can swap the 1-st and the 3-rd elements to get $[3, 2, 1]$;
- if we shift $[1, 2, 3]$ by 1 position, we get $[3, 1, 2]$. Then we can swap the 2-nd and the 3-rd elements to get $[3, 2, 1]$;
- if we shift $[1, 2, 3]$ by 2 positions, we get $[2, 3, 1]$. Then we can swap the 1-st and the 2-nd elements to get $[3, 2, 1]$;

- in the fourth test case, we stated that we didn't do any swaps after the cyclic shift, but no value of cyclic shift could produce the array $[1, 2, 3, 4, 6, 5]$.

F. Pairwise Modulo

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have an array a consisting of n distinct positive integers, numbered from 1 to n. Define p_k as

$$p_k = \sum_{1 \le i, j \le k} a_i \bmod a_j,$$

where $x \bmod y$ denotes the remainder when x is divided by y. You have to find and print $p_1, p_2, \ldots, p_n$.

Input
The first line contains n — the length of the array ($2 \le n \le 2 \cdot 10^5$).

The second line contains n space-separated distinct integers $a_1, \ldots, a_n$ ($1 \le a_i \le 3 \cdot 10^5$, $a_i \ne a_j$ if $i \ne j$).

Output
Print n integers $p_1, p_2, \ldots, p_n$.

Examples

input
4 6 2 7 3
output
0 2 12 22

input
3 3 2 1
output
0 3 5

G. Common Divisor Graph

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Consider a sequence of distinct integers $a_1, \ldots, a_n$, each representing one node of a graph. There is an edge between two nodes if the two values are not coprime, i. e. they have a common divisor greater than 1.

There are q queries, in each query, you want to get from one given node a_s to another a_t. In order to achieve that, you can choose an existing value a_i and create new value $a_{n+1} = a_i \cdot (1 + a_i)$, with edges to all values that are not coprime with a_{n+1}. Also, n gets increased by 1. You can repeat that operation multiple times, possibly making the sequence much longer and getting huge or repeated values. What's the minimum possible number of newly created nodes so that a_t is reachable from a_s?

Queries are independent. In each query, you start with the initial sequence a given in the input.

Input
The first line contains two integers n and q ($2 \le n \le 150\,000$, $1 \le q \le 300\,000$) — the size of the sequence and the number of queries.

The second line contains n **distinct** integers $a_1, a_2, \ldots, a_n$ ($2 \le a_i \le 10^6$, $a_i \ne a_j$ if $i \ne j$).

The j-th of the following q lines contains two distinct integers s_j and t_j ($1 \le s_j, t_j \le n$, $s_j \ne t_j$) — indices of nodes for j-th query.

Output
Print q lines. The j-th line should contain one integer: the minimum number of new nodes you create in order to move from a_{s_j} to a_{t_j}.

Examples

input
3 3 2 10 3 1 2 1 3 2 3

output
0 1 1

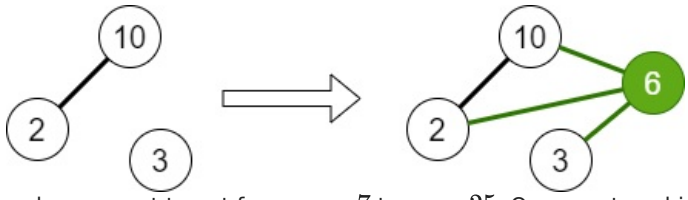
input
5 12 3 8 7 6 25 1 2 1 3 1 4 1 5 2 1 2 3 2 4 2 5 3 1 3 2 3 4 3 5

output
0 1 0 1 0 1 0 1 1 1 1 2

Note

In the first example, you can first create new value $2 \cdot 3 = 6$ or $10 \cdot 11 = 110$ or $3 \cdot 4 = 12$. None of that is needed in the first query because you can already get from $a_1 = 2$ to $a_2 = 10$.

In the second query, it's optimal to first create 6 or 12. For example, creating 6 makes it possible to get from $a_1 = 2$ to $a_3 = 3$ with a path $(2, 6, 3)$.



In the last query of the second example, we want to get from $a_3 = 7$ to $a_5 = 25$. One way to achieve that is to first create $6 \cdot 7 = 42$ and then create $25 \cdot 26 = 650$. The final graph has seven nodes and it contains a path from $a_3 = 7$ to $a_5 = 25$.

H. XOR and Distance

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an array a consisting of n distinct elements and an integer k. Each element in the array is a non-negative integer not exceeding $2^k - 1$.

Let's define the *XOR distance* for a number x as the value of

$$f(x) = \min_{i=1}^{n} \min_{j=i+1}^{n} |(a_i \oplus x) - (a_j \oplus x)|,$$

where $\oplus$ denotes the bitwise XOR operation.

For every integer x from 0 to $2^k - 1$, you have to calculate $f(x)$.

Input

The first line contains two integers n and k ($1 \le k \le 19$; $2 \le n \le 2^k$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 2^k - 1$). All these integers are distinct.

Output
Print 2^k integers. The i-th of them should be equal to $f(i-1)$.

Examples

input
3 3 6 0 3
output
3 1 1 2 2 1 1 3

input
3 4 13 4 2
output
2 2 6 6 3 1 2 2 2 2 1 3 6 6 2 2

Note
Consider the first example:

- for $x = 0$, if we apply bitwise XOR to the elements of the array with x, we get the array $[6, 0, 3]$, and the minimum absolute difference of two elements is 3;
- for $x = 1$, if we apply bitwise XOR to the elements of the array with x, we get the array $[7, 1, 2]$, and the minimum absolute difference of two elements is 1;
- for $x = 2$, if we apply bitwise XOR to the elements of the array with x, we get the array $[4, 2, 1]$, and the minimum absolute difference of two elements is 1;
- for $x = 3$, if we apply bitwise XOR to the elements of the array with x, we get the array $[5, 3, 0]$, and the minimum absolute difference of two elements is 2;
- for $x = 4$, if we apply bitwise XOR to the elements of the array with x, we get the array $[2, 4, 7]$, and the minimum absolute difference of two elements is 2;
- for $x = 5$, if we apply bitwise XOR to the elements of the array with x, we get the array $[3, 5, 6]$, and the minimum absolute difference of two elements is 1;
- for $x = 6$, if we apply bitwise XOR to the elements of the array with x, we get the array $[0, 6, 5]$, and the minimum absolute difference of two elements is 1;
- for $x = 7$, if we apply bitwise XOR to the elements of the array with x, we get the array $[1, 7, 4]$, and the minimum absolute difference of two elements is 3.

I. Stairs

time limit per test: 10 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

For a permutation p of numbers 1 through n, we define a *stair array* a as follows: a_i is length of the longest segment of permutation which contains position i and is made of consecutive values in sorted order: $[x, x+1, \ldots, y-1, y]$ or $[y, y-1, \ldots, x+1, x]$ for some $x \le y$. For example, for permutation $p = [4, 1, 2, 3, 7, 6, 5]$ we have $a = [1, 3, 3, 3, 3, 3, 3]$.

You are given the stair array a. Your task is to calculate the number of permutations which have stair array equal to a. Since the number can be big, compute it modulo $998\,244\,353$. Note that this number can be equal to zero.

Input
The first line of input contains integer n ($1 \le n \le 10^5$) — the length of a stair array a.

The second line of input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

Output
Print the number of permutations which have stair array equal to a. Since the number can be big, compute it modulo $998\,244\,353$.

Examples

input
6 3 3 3 1 1 1
output
6

input
7 4 4 4 4 3 3 3
output

6

input
1 1
output
1

input
8 2 2 2 2 2 2 1 1
output
370

input
4 3 2 3 1
output
0