Codeforces Round #598 (Div. 3)

A. Payment Without Change

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

You have a coins of value n and b coins of value 1. You always pay in exact change, so you want to know if there exist such x and y that if you take x ($0 \le x \le a$) coins of value n and y ($0 \le y \le b$) coins of value 1, then the total value of taken coins will be S.

You have to answer q independent test cases.

Input

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

The only line of the test case contains four integers a, b, n and S ($1 \le a, b, n, S \le 10^9$) — the number of coins of value n, the number of coins of value 1, the value n and the required total value.

Output

For the i-th test case print the answer on it — YES (without quotes) if there exist such x and y that if you take x coins of value n and y coins of value 1, then the total value of taken coins will be S, and NO otherwise.

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes and YES will all be recognized as positive answer).

Example

input
4 1 2 3 4 1 2 3 6 5 2 6 27 3 3 5 18
output
YES NO NO YES

B. Minimize the Permutation

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

You are given a permutation of length n. Recall that the permutation is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

You can perform at most $n - 1$ operations with the given permutation (it is possible that you don't perform any operations at all). The i-th operation allows you to swap elements of the given permutation on positions i and $i + 1$. **Each operation can be performed at most once**. The operations can be performed in arbitrary order.

Your task is to find the lexicographically minimum possible permutation obtained by performing some of the given operations in some order.

You can see the definition of the lexicographical order in the notes section.

You have to answer q independent test cases.

For example, let's consider the permutation $[5, 4, 1, 3, 2]$. The minimum possible permutation we can obtain is $[1, 5, 2, 4, 3]$ and we can do it in the following way:

1. perform the second operation (swap the second and the third elements) and obtain the permutation $[5, 1, 4, 3, 2]$;
2. perform the fourth operation (swap the fourth and the fifth elements) and obtain the permutation $[5, 1, 4, 2, 3]$;
3. perform the third operation (swap the third and the fourth elements) and obtain the permutation $[5, 1, 2, 4, 3]$.
4. perform the first operation (swap the first and the second elements) and obtain the permutation $[1, 5, 2, 4, 3]$;

Another example is $[1, 2, 4, 3]$. The minimum possible permutation we can obtain is $[1, 2, 3, 4]$ by performing the third operation (swap the third and the fourth elements).

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

The first line of the test case contains one integer n ($1 \le n \le 100$) — the number of elements in the permutation.

The second line of the test case contains n distinct integers from 1 to n — the given permutation.

Output
For each test case, print the answer on it — the lexicograhically minimum possible permutation obtained by performing some of the given operations in some order.

Example

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

Note
Recall that the permutation p of length n is lexicographically less than the permutation q of length n if there is such index $i \le n$ that for all j from 1 to $i - 1$ the condition $p_j = q_j$ is satisfied, and $p_i < q_i$. For example:

- $p = [1, 3, 5, 2, 4]$ is less than $q = [1, 3, 5, 4, 2]$ (such $i = 4$ exists, that $p_i < q_i$ and for each $j < i$ holds $p_j = q_j$),
- $p = [1, 2]$ is less than $q = [2, 1]$ (such $i = 1$ exists, that $p_i < q_i$ and for each $j < i$ holds $p_j = q_j$).

C. Platforms Jumping

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

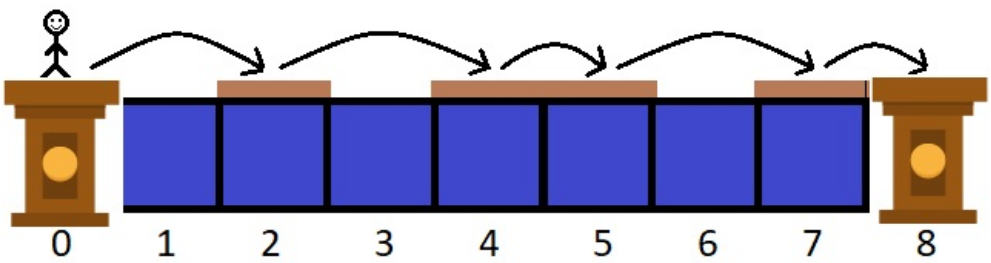
There is a river of width n. The left bank of the river is cell 0 and the right bank is cell $n + 1$ (more formally, the river can be represented as a sequence of $n + 2$ cells numbered from 0 to $n + 1$). There are also m wooden platforms on a river, the i-th platform has length c_i (so the i-th platform takes c_i consecutive cells of the river). It is guaranteed that the sum of lengths of platforms does not exceed n.

You are standing at 0 and want to reach $n + 1$ somehow. If you are standing at the position x, you can jump to any position in the range $[x + 1; x + d]$. **However** you don't really like the water so you can jump only to such cells that belong to some wooden platform. For example, if $d = 1$, you can jump only to the next position (if it belongs to the wooden platform). **You can assume that cells 0 and $n + 1$ belong to wooden platforms**.

You want to know if it is possible to reach $n + 1$ from 0 if you can move any platform to the left or to the right arbitrary number of times (possibly, zero) as long as they do not intersect each other (but two platforms can touch each other). It also means that you cannot change the relative order of platforms.

Note that you should move platforms until you start jumping (in other words, you first move the platforms and then start jumping).

For example, if $n = 7$, $m = 3$, $d = 2$ and $c = [1, 2, 1]$, then one of the ways to reach 8 from 0 is follow:



The first example: $n = 7$.

Input

The first line of the input contains three integers n, m and d ($1 \le n, m, d \le 1000, m \le n$) — the width of the river, the number of platforms and the maximum distance of your jump, correspondingly.

The second line of the input contains m integers $c_1, c_2, \ldots, c_m$ ($1 \le c_i \le n, \sum\limits_{i=1}^{m} c_i \le n$), where c_i is the length of the i-th platform.

Output

If it is impossible to reach $n + 1$ from 0, print NO in the first line. Otherwise, print YES in the first line and the array a of length n in the second line — the sequence of river cells (excluding cell 0 and cell $n + 1$).

If the cell i does not belong to any platform, a_i should be 0. Otherwise, it should be equal to the index of the platform (1-indexed, platforms are numbered from 1 to m in order of input) to which the cell i belongs.

Note that all a_i equal to 1 should form a contiguous subsegment of the array a of length c_1, all a_i equal to 2 should form a contiguous subsegment of the array a of length c_2, ..., all a_i equal to m should form a contiguous subsegment of the array a of length c_m. The leftmost position of 2 in a should be greater than the rightmost position of 1, the leftmost position of 3 in a should be greater than the rightmost position of 2, ..., the leftmost position of m in a should be greater than the rightmost position of $m - 1$.

See example outputs for better understanding.

Examples

input
7 3 2 1 2 1
output
YES 0 1 0 2 2 0 3

input
10 1 11 1
output
YES 0 0 0 0 0 0 0 0 0 1

input
10 1 5 2
output
YES 0 0 0 0 1 1 0 0 0 0

Note

Consider the first example: the answer is $[0, 1, 0, 2, 2, 0, 3]$. The sequence of jumps you perform is $0 \to 2 \to 4 \to 5 \to 7 \to 8$.

Consider the second example: it does not matter how to place the platform because you always can jump from 0 to 11.

Consider the third example: the answer is $[0, 0, 0, 0, 1, 1, 0, 0, 0, 0]$. The sequence of jumps you perform is $0 \to 5 \to 6 \to 11$.

D. Binary String Minimizing

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

You are given a binary string of length n (i. e. a string consisting of n characters '0' and '1').

In one move you can swap two adjacent characters of the string. What is the lexicographically minimum possible string you can obtain from the given one if you can perform **no more** than k moves? It is possible that you do not perform any moves at all.

Note that you can swap the same pair of adjacent characters with indices i and $i + 1$ arbitrary (possibly, zero) number of times. Each such swap is considered a separate move.

You have to answer q independent test cases.

Input

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

The first line of the test case contains two integers n and k ($1 \le n \le 10^6, 1 \le k \le n^2$) — the length of the string and the number of moves you can perform.

The second line of the test case contains one string consisting of n characters '0' and '1'.

It is guaranteed that the sum of n over all test cases does not exceed 10^6 ($\sum n \le 10^6$).

Output
For each test case, print the answer on it: the lexicographically minimum possible string of length n you can obtain from the given one if you can perform **no more** than k moves.

Example

input
3 8 5 11011010 7 9 1111100 7 11 1111100
output
01011110 0101111 0011111

Note
In the first example, you can change the string as follows:
$1\underline{10}11010 \to \underline{10}111010 \to 0111\underline{10}10 \to 01\underline{110}110 \to 01\underline{10}1110 \to 01011110$.

In the third example, there are enough operations to make the string sorted.

E. Yet Another Division Into Teams

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

There are n students at your university. The programming skill of the i-th student is a_i. As a coach, you want to divide them into teams to prepare them for the upcoming ICPC finals. Just imagine how good this university is if it has $2 \cdot 10^5$ students ready for the finals!

Each team should consist of **at least three students**. Each student should belong to **exactly one team**. The *diversity* of a team is the difference between the **maximum** programming skill of some student that belongs to this team and the **minimum** programming skill of some student that belongs to this team (in other words, if the team consists of k students with programming skills $a[i_1], a[i_2], \ldots, a[i_k]$, then the *diversity* of this team is $\max\limits_{j=1}^{k} a[i_j] - \min\limits_{j=1}^{k} a[i_j]$).

The total *diversity* is the sum of *diversities* of all teams formed.

Your task is to minimize the total *diversity* of the division of students and find the optimal way to divide the students.

Input
The first line of the input contains one integer n ($3 \le n \le 2 \cdot 10^5$) — the number of students.

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

Output
In the first line print two integers res and k — the minimum total *diversity* of the division of students and the number of teams in your division, correspondingly.

In the second line print n integers $t_1, t_2, \ldots, t_n$ ($1 \le t_i \le k$), where t_i is the number of team to which the i-th student belong.

If there are multiple answers, you can print any. Note that you don't need to minimize the number of teams. Each team should consist of **at least three students**.

Examples

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

input
6 1 5 12 13 2 15
output

```
7 2
2 2 1 1 2 1
```

input
10 1 2 5 129 185 581 1041 1909 1580 8150
output
7486 3 3 3 3 2 2 2 2 1 1 1

Note

In the first example, there is only one team with skills $[1, 1, 2, 3, 4]$ so the answer is 3. It can be shown that you cannot achieve a better answer.

In the second example, there are two teams with skills $[1, 2, 5]$ and $[12, 13, 15]$ so the answer is $4 + 3 = 7$.

In the third example, there are three teams with skills $[1, 2, 5]$, $[129, 185, 581, 1041]$ and $[1580, 1909, 8150]$ so the answer is $4 + 912 + 6570 = 7486$.

F. Equalizing Two Strings

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

You are given two strings s and t both of length n and both consisting of lowercase Latin letters.

In one move, you can choose any length len from 1 to n and perform the following operation:

- Choose any contiguous substring of the string s of length len and reverse it;
- **at the same time** choose any contiguous substring of the string t of length len and reverse it as well.

Note that during one move you reverse **exactly one** substring of the string s and **exactly one** substring of the string t.

Also note that borders of substrings you reverse in s and in t **can be different**, the only restriction is that you reverse the substrings of equal length. For example, if $len = 3$ and $n = 5$, you can reverse $s[1 \dots 3]$ and $t[3 \dots 5]$, $s[2 \dots 4]$ and $t[2 \dots 4]$, but not $s[1 \dots 3]$ and $t[1 \dots 2]$.

Your task is to say if it is possible to make strings s and t equal after some (possibly, empty) sequence of moves.

You have to answer q independent test cases.

Input

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

The first line of the test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the length of s and t.

The second line of the test case contains one string s consisting of n lowercase Latin letters.

The third line of the test case contains one string t consisting of n lowercase Latin letters.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output

For each test case, print the answer on it — "YES" (without quotes) if it is possible to make strings s and t equal after some (possibly, empty) sequence of moves and "NO" otherwise.

Example

input
4 4 abcd abdc 5 ababa baaba 4 asdf asdg 4 abcd badc
output
NO YES NO YES