Codeforces Round #704 (Div. 2)

A. Three swimmers

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Three swimmers decided to organize a party in the swimming pool! At noon, they started to swim from the left side of the pool.

It takes the first swimmer exactly a minutes to swim across the entire pool and come back, exactly b minutes for the second swimmer and c minutes for the third. Hence, the first swimmer will be on the left side of the pool after 0, a, $2a$, $3a$, ... minutes after the start time, the second one will be at 0, b, $2b$, $3b$, ... minutes, and the third one will be on the left side of the pool after 0, c, $2c$, $3c$, ... minutes.

You came to the left side of the pool exactly p minutes after they started swimming. Determine how long you have to wait before one of the swimmers arrives at the left side of the pool.

Input

The first line of the input contains a single integer t ($1 \le t \le 1000$) — the number of test cases. Next t lines contains test case descriptions, one per line.

Each line contains four integers p, a, b and c ($1 \le p, a, b, c \le 10^{18}$), time in minutes after the start, when you came to the pool and times in minutes it take the swimmers to cross the entire pool and come back.

Output

For each test case, output one integer — how long you have to wait (in minutes) before one of the swimmers arrives at the left side of the pool.

Example

input
4 9 5 4 8 2 6 10 9 10 2 5 10 10 9 9 9
output
1 4 0 8

Note

In the first test case, the first swimmer is on the left side in $0, 5, 10, 15, \ldots$ minutes after the start time, the second swimmer is on the left side in $0, 4, 8, 12, \ldots$ minutes after the start time, and the third swimmer is on the left side in $0, 8, 16, 24, \ldots$ minutes after the start time. You arrived at the pool in 9 minutes after the start time and in a minute you will meet the first swimmer on the left side.

In the second test case, the first swimmer is on the left side in $0, 6, 12, 18, \ldots$ minutes after the start time, the second swimmer is on the left side in $0, 10, 20, 30, \ldots$ minutes after the start time, and the third swimmer is on the left side in $0, 9, 18, 27, \ldots$ minutes after the start time. You arrived at the pool 2 minutes after the start time and after 4 minutes meet the first swimmer on the left side.

In the third test case, you came to the pool 10 minutes after the start time. At the same time, all three swimmers are on the left side. A rare stroke of luck!

In the fourth test case, all swimmers are located on the left side in $0, 9, 18, 27, \ldots$ minutes after the start time. You arrived at the pool 10 minutes after the start time and after 8 minutes meet all three swimmers on the left side.

B. Card Deck

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

You have a deck of n cards, and you'd like to reorder it to a new one.

Each card has a value between 1 and n equal to p_i. All p_i are pairwise distinct. Cards in a deck are numbered from bottom to top, i. e. p_1 stands for the bottom card, p_n is the top card.

In each step you pick some integer $k > 0$, take the top k cards from the original deck and place them, in the order they are now, on top of the new deck. You perform this operation until the original deck is empty. (Refer to the notes section for the better understanding.)

Let's define an *order of a deck* as $\sum_{i=1}^{n} n^{n-i} \cdot p_i$.

Given the original deck, output the deck with maximum possible order you can make using the operation above.

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 ($1 \le n \le 10^5$) — the size of deck you have.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$; $p_i \neq p_j$ if $i \neq j$) — values of card in the deck from bottom to top.

It's guaranteed that the sum of n over all test cases doesn't exceed 10^5.

Output
For each test case print the deck with maximum possible order. Print values of cards in the deck from bottom to top.

If there are multiple answers, print any of them.

Example

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

Note
In the first test case, one of the optimal strategies is the next one:

1. take 1 card from the top of p and move it to p': p becomes $[1, 2, 3]$, p' becomes $[4]$;
2. take 1 card from the top of p: p becomes $[1, 2]$, p' becomes $[4, 3]$;
3. take 1 card from the top of p: p becomes $[1]$, p' becomes $[4, 3, 2]$;
4. take 1 card from the top of p: p becomes empty, p' becomes $[4, 3, 2, 1]$.

In result, p' has order equal to $4^3 \cdot 4 + 4^2 \cdot 3 + 4^1 \cdot 2 + 4^0 \cdot 1 = 256 + 48 + 8 + 1 = 313$.
In the second test case, one of the optimal strategies is:

1. take 4 cards from the top of p and move it to p': p becomes $[1]$, p' becomes $[5, 2, 4, 3]$;
2. take 1 card from the top of p and move it to p': p becomes empty, p' becomes $[5, 2, 4, 3, 1]$;

In result, p' has order equal to $5^4 \cdot 5 + 5^3 \cdot 2 + 5^2 \cdot 4 + 5^1 \cdot 3 + 5^0 \cdot 1 = 3125 + 250 + 100 + 15 + 1 = 3491$.
In the third test case, one of the optimal strategies is:

1. take 2 cards from the top of p and move it to p': p becomes $[4, 2, 5, 3]$, p' becomes $[6, 1]$;
2. take 2 cards from the top of p and move it to p': p becomes $[4, 2]$, p' becomes $[6, 1, 5, 3]$;
3. take 2 cards from the top of p and move it to p': p becomes empty, p' becomes $[6, 1, 5, 3, 4, 2]$.

In result, p' has order equal to $6^5 \cdot 6 + 6^4 \cdot 1 + 6^3 \cdot 5 + 6^2 \cdot 3 + 6^1 \cdot 4 + 6^0 \cdot 2 = 46656 + 1296 + 1080 + 108 + 24 + 2 = 49166$.

C. Maximum width

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

Your classmate, whom you do not like because he is boring, but whom you respect for his intellect, has two strings: s of length n and t of length m.

A sequence $p_1, p_2, \ldots, p_m$, where $1 \le p_1 < p_2 < \ldots < p_m \le n$, is called *beautiful*, if $s_{p_i} = t_i$ for all i from 1 to m. The *width* of a sequence is defined as $\max\limits_{1 \le i < m} (p_{i+1} - p_i)$.

Please help your classmate to identify the beautiful sequence with the **maximum width**. Your classmate promised you that for the given strings s and t there is at least one beautiful sequence.

Input

The first input line contains two integers n and m ($2 \le m \le n \le 2 \cdot 10^5$) — the lengths of the strings s and t.

The following line contains a single string s of length n, consisting of lowercase letters of the Latin alphabet.

The last line contains a single string t of length m, consisting of lowercase letters of the Latin alphabet.

It is guaranteed that there is at least one beautiful sequence for the given strings.

Output

Output one integer — the maximum width of a beautiful sequence.

Examples

input
5 3 abbbc abc
output
3

input
5 2 aaaaa aa
output
4

input
5 5 abcdf abcdf
output
1

input
2 2 ab ab
output
1

Note

In the first example there are two beautiful sequences of width 3: they are $\{1, 2, 5\}$ and $\{1, 4, 5\}$.

In the second example the beautiful sequence with the maximum width is $\{1, 5\}$.

In the third example there is exactly one beautiful sequence — it is $\{1, 2, 3, 4, 5\}$.

In the fourth example there is exactly one beautiful sequence — it is $\{1, 2\}$.

D. Genius's Gambit

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

You are given three integers a, b, k.

Find two *binary* integers x and y ($x \ge y$) such that

1. both x and y consist of a zeroes and b ones;
2. $x - y$ (also written in binary form) has exactly k ones.

You are **not allowed to use leading zeros for x and y**.

Input

The only line contains three integers a, b, and k ($0 \le a$; $1 \le b$; $0 \le k \le a + b \le 2 \cdot 10^5$) — the number of zeroes, ones, and the number of ones in the result.

Output
If it's possible to find two suitable integers, print "Yes" followed by x and y in base-2.

Otherwise print "No".

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

Examples

input
4 2 3
output
Yes 101000 100001

input
3 2 1
output
Yes 10100 10010

input
3 2 5
output
No

Note
In the first example, $x = 101000_2 = 2^5 + 2^3 = 40_{10}$, $y = 100001_2 = 2^5 + 2^0 = 33_{10}$, $40_{10} - 33_{10} = 7_{10} = 2^2 + 2^1 + 2^0 = 111_2$. Hence $x - y$ has 3 ones in base-2.

In the second example, $x = 10100_2 = 2^4 + 2^2 = 20_{10}$, $y = 10010_2 = 2^4 + 2^1 = 18$, $x - y = 20 - 18 = 2_{10} = 10_2$. This is precisely one 1.

In the third example, one may show, that it's impossible to find an answer.

E. Almost Fault-Tolerant Database

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

You are storing an integer array of length m in a database. To maintain internal integrity and protect data, the database stores n copies of this array.

Unfortunately, the recent incident may have altered the stored information in every copy in the database.

It's believed, that the incident altered at most two elements in every copy. You need to recover the original array based on the current state of the database.

In case there are multiple ways to restore the array, report any. If there is no array that differs from every copy in no more than two positions, report that as well.

Input
The first line contains integers n and m ($2 \le n$; $1 \le m$; $n \cdot m \le 250\,000$) — the number of copies and the size of the array.

Each of the following n lines describes one of the currently stored copies in the database, it consists of m integers $s_{i,1}, s_{i,2}, \ldots, s_{i,m}$ ($1 \le s_{i,j} \le 10^9$).

Output
If there is an array consistent with all given copies, print "Yes" and then the array itself. The array must have length m and contain integers between 1 and 10^9 only.

Otherwise, print "No".

If there are multiple possible arrays, print any of them.

Examples

input
3 4 1 10 10 100 1 1 1 100

10 100 1 100

output

Yes
1 10 1 100

input

10 7
1 1 1 1 1 1 1
1 1 1 1 1 1 2
1 1 1 1 1 2 2
1 1 1 1 2 2 1
1 1 1 2 2 1 1
1 1 2 2 1 1 1
1 2 2 1 1 1 1
2 2 1 1 1 1 1
2 1 1 1 1 1 1
1 1 1 1 1 1 1

output

Yes
1 1 1 1 1 1 1

input

2 5
2 2 1 1 1
1 1 2 2 2

output

No

Note

In the first example, the array $[1, 10, 1, 100]$ differs from first and second copies in just one position, and from the third copy in two positions.

In the second example, array $[1, 1, 1, 1, 1, 1, 1]$ is the same as the first copy and differs from all other copies in at most two positions.

In the third example, there is no array differing in at most two positions from every database's copy.