Intel Code Challenge Elimination Round (Div.1 + Div.2, combined)

A. Broken Clock

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

You are given a broken clock. You know, that it is supposed to show time in 12- or 24-hours `HH:MM` format. In 12-hours format hours change from 1 to 12, while in 24-hours it changes from 0 to 23. In both formats minutes change from 0 to 59.

You are given a time in format `HH:MM` that is currently displayed on the broken clock. Your goal is to change minimum number of digits in order to make clocks display the correct time in the given format.

For example, if `00:99` is displayed, it is enough to replace the second `9` with `3` in order to get `00:39` that is a correct time in 24-hours format. However, to make `00:99` correct in 12-hours format, one has to change at least two digits. Additionally to the first change one can replace the second `0` with `1` and obtain `01:39`.

Input

The first line of the input contains one integer 12 or 24, that denote 12-hours or 24-hours format respectively.

The second line contains the time in format `HH:MM`, that is currently displayed on the clock. First two characters stand for the hours, while next two show the minutes.

Output

The only line of the output should contain the time in format `HH:MM` that is a correct time in the given format. It should differ from the original in as few positions as possible. If there are many optimal solutions you can print any of them.

Examples

input
24 17:30
output
17:30

input
12 17:30
output
07:30

input
24 99:99
output
09:09

B. Verse Pattern

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

You are given a text consisting of n lines. Each line contains some space-separated words, consisting of lowercase English letters.

We define a syllable as a string that contains exactly one vowel and any arbitrary number (possibly none) of consonants. In English alphabet following letters are considered to be vowels: 'a', 'e', 'i', 'o', 'u' and 'y'.

Each word of the text that contains at least one vowel can be divided into syllables. Each character should be a part of exactly one syllable. For example, the word "mamma" can be divided into syllables as "ma" and "mma", "mam" and "ma", and "mamm" and "a". Words that consist of only consonants should be ignored.

The verse patterns for the given text is a sequence of n integers $p_1, p_2, ..., p_n$. Text matches the given verse pattern if for each i from 1 to n one can divide words of the i-th line in syllables in such a way that the total number of syllables is equal to p_i.

You are given the text and the verse pattern. Check, if the given text matches the given verse pattern.

Input

The first line of the input contains a single integer n $(1 \le n \le 100)$ — the number of lines in the text.

The second line contains integers $p_1, ..., p_n$ $(0 \le p_i \le 100)$ — the verse pattern.

Next n lines contain the text itself. Text consists of lowercase English letters and spaces. It's guaranteed that all lines are non-empty, each line starts and ends with a letter and words are separated by exactly one space. The length of each line doesn't exceed 100 characters.

Output

If the given text matches the given verse pattern, then print "YES" (without quotes) in the only line of the output. Otherwise, print "NO" (without quotes).

Examples

input
3 2 2 3 intel code ch allenge
output
YES

input
4 1 2 3 1 a bcdefghi jklmnopqrstu vwxyz
output
NO

input
4 13 11 15 15 to be or not to be that is the question whether tis nobler in the mind to suffer the slings and arrows of outrageous fortune or to take arms against a sea of troubles
output
YES

Note

In the first sample, one can split words into syllables in the following way:

```
in-tel
co-de
ch al-len-ge
```

Since the word "ch" in the third line doesn't contain vowels, we can ignore it. As the result we get 2 syllabels in first two lines and 3 syllables in the third one.

C. Destroying Array

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

You are given an array consisting of n non-negative integers $a_1, a_2, ..., a_n$.

You are going to destroy integers in the array one by one. Thus, you are given the permutation of integers from 1 to n defining the order elements of the array are destroyed.

After each element is destroyed you have to find out the segment of the array, such that it contains no destroyed elements and the sum of its elements is maximum possible. The sum of elements in the empty segment is considered to be 0.

Input

The first line of the input contains a single integer n $(1 \le n \le 100\,000)$ — the length of the array.

The second line contains n integers $a_1, a_2, ..., a_n$ $(0 \le a_i \le 10^9)$.

The third line contains a permutation of integers from 1 to n — the order used to destroy elements.

Output

Print n lines. The i-th line should contain a single integer — the maximum possible sum of elements on the segment containing no destroyed elements, after first i operations are performed.

Examples

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

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

input
8 5 5 4 4 6 6 5 5 5 2 8 7 1 3 4 6
output
18 16 11 8 8 6 6 0

Note

Consider the first sample:

1. Third element is destroyed. Array is now $1\ 3\ *\ 5$. Segment with maximum sum 5 consists of one integer 5.
2. Fourth element is destroyed. Array is now $1\ 3\ *\ *$. Segment with maximum sum 4 consists of two integers $1\ 3$.
3. First element is destroyed. Array is now $*\ 3\ *\ *$. Segment with maximum sum 3 consists of one integer 3.
4. Last element is destroyed. At this moment there are no valid nonempty segments left in this array, so the answer is equal to 0.

D. Generating Sets

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

You are given a set Y of n **distinct** positive integers $y_1, y_2, ..., y_n$.

Set X of n **distinct** positive integers $x_1, x_2, ..., x_n$ is said to *generate* set Y if one can transform X to Y by applying some number of the following two operation to integers in X:

1. Take any integer x_i and multiply it by two, i.e. replace x_i with $2 \cdot x_i$.
2. Take any integer x_i, multiply it by two and add one, i.e. replace x_i with $2 \cdot x_i + 1$.

Note that integers in X are not required to be distinct after each operation.

Two sets of distinct integers X and Y are equal if they are equal as sets. In other words, if we write elements of the sets in the array in the increasing order, these arrays would be equal.

Note, that any set of integers (or its permutation) generates itself.

You are given a set Y and have to find a set X that generates Y and the **maximum element of X is mininum possible**.

Input

The first line of the input contains a single integer n ($1 \le n \le 50\,000$) — the number of elements in Y.

The second line contains n integers $y_1, ..., y_n$ ($1 \le y_i \le 10^9$), that are guaranteed to be distinct.

Output

Print n integers — set of distinct integers that generate Y and the maximum element of which is minimum possible. If there are several such sets, print any of them.

Examples

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

input
6 15 14 3 13 1 12
output
12 13 14 7 3 1

input
6 9 7 13 17 5 11
output
4 5 2 6 3 1

E. Research Rover

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Unfortunately, the formal description of the task turned out to be too long, so here is the legend.

Research rover finally reached the surface of Mars and is ready to complete its mission. Unfortunately, due to the mistake in the navigation system design, the rover is located in the wrong place.

The rover will operate on the grid consisting of n rows and m columns. We will define as (r, c) the cell located in the row r and column c. From each cell the rover is able to move to any cell that share a side with the current one.

The rover is currently located at cell $(1, 1)$ and has to move to the cell (n, m). It will randomly follow some **shortest path** between these two cells. Each possible way is chosen equiprobably.

The cargo section of the rover contains the battery required to conduct the research. Initially, the battery charge is equal to s units of energy.

Some of the cells contain anomaly. Each time the rover gets to the cell with anomaly, the battery looses half of its charge rounded down. Formally, if the charge was equal to x before the rover gets to the cell with anomaly, the charge will change to .

While the rover picks a random shortest path to proceed, compute the expected value of the battery charge after it reaches cell (n, m). If the cells $(1, 1)$ and (n, m) contain anomaly, they also affect the charge of the battery.

Input

The first line of the input contains four integers n, m, k and s ($1 \le n, m \le 100\,000$, $0 \le k \le 2000$, $1 \le s \le 1\,000\,000$) — the number of rows and columns of the field, the number of cells with anomaly and the initial charge of the battery respectively.

The follow k lines containing two integers r_i and c_i ($1 \le r_i \le n$, $1 \le c_i \le m$) — coordinates of the cells, containing anomaly. It's guaranteed that each cell appears in this list no more than once.

Output

The answer can always be represented as an irreducible fraction . Print the only integer $P \cdot Q^{-1}$ modulo $10^9 + 7$.

Examples

input
3 3 2 11 2 1 2 3

output
333333342

input
4 5 3 17 1 2 3 3 4 1

output
514285727

input
1 6 2 15 1 1 1 5

output
4

Note

In the first sample, the rover picks one of the following six routes:

1. , after passing cell $(2, 3)$ charge is equal to 6.
2. , after passing cell $(2, 3)$ charge is equal to 6.
3. , charge remains unchanged and equals 11.
4. , after passing cells $(2, 1)$ and $(2, 3)$ charge equals 6 and then 3.
5. , after passing cell $(2, 1)$ charge is equal to 6.
6. , after passing cell $(2, 1)$ charge is equal to 6.

Expected value of the battery charge is calculated by the following formula:

.

Thus $P = 19$, and $Q = 3$.

3^{-1} modulo $10^9 + 7$ equals 333333336.

$19 \cdot 333333336 = 333333342 \ (mod\ 10^9 + 7)$

F. Cyclic Cipher

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

You are given n sequences. Each sequence consists of positive integers, not exceeding m. All integers in one sequence are distinct, but the same integer may appear in multiple sequences. The length of the i-th sequence is k_i.

Each second integers in each of the sequences are shifted by one to the left, i.e. integers at positions $i > 1$ go to positions i - 1, while the first integers becomes the last.

Each second we take the first integer of each sequence and write it down to a new array. Then, for each value x from 1 to m we compute the longest **segment** of the array consisting of element x only.

The above operation is performed for 10^{100} seconds. For each integer from 1 to m find out the longest segment found at this time.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 100\,000$) — the number of sequences and the maximum integer that can appear in the sequences.

Then follow n lines providing the sequences. Each of them starts with an integer k_i ($1 \le k_i \le 40$) — the number of integers in the sequence, proceeded by k_i positive integers — elements of the sequence. It's guaranteed that all integers in each sequence are pairwise distinct and do not exceed m.

The total length of all sequences doesn't exceed $200\,000$.

Output

Print m integers, the i-th of them should be equal to the length of the longest segment of the array with all its values equal to i during the first 10^{100} seconds.

Examples

input

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

output

```
2
1
3
2
```

input

```
5 5
2 3 1
4 5 1 3 2
4 2 1 3 5
1 3
2 5 3
```

output

```
3
1
4
0
1
```

input

```
4 6
3 4 5 3
2 6 3
2 3 6
3 3 6 5
```

output

```
0
0
2
1
1
2
```