A. Everything Nim

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Alice and Bob are playing a game on n piles of stones. On each player's turn, they select a positive integer k that is at most the size of the smallest **nonempty** pile and remove k stones from **each** nonempty pile at once. The first player who is unable to make a move (because all piles are empty) loses.

Given that Alice goes first, who will win the game if both players play optimally?

Input

The first line of the input contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of piles in the game.

The next line of each test case contains n integers $a_1, a_2, \ldots a_n$ ($1 \leq a_i \leq 10^9$), where a_i is the initial number of stones in the i-th pile.

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

Output

For each test case, print a single line with the name of the winner, assuming both players play optimally. If Alice wins, print "`Alice`", otherwise print "`Bob`" (without quotes).

Standard Input	Standard Output
7 5 3 3 3 3 3 2 1 7 7 1 3 9 7 4 2 100 3 1 2 3 6 2 1 3 4 2 4 8 5 7 2 9 6 3 3 2 1 1000000000	Alice Bob Alice Alice Bob Alice Alice

Note

In the first test case, Alice can win by choosing $k = 3$ on her first turn, which will empty all of the piles at once.

In the second test case, Alice must choose $k = 1$ on her first turn since there is a pile of size 1, so Bob can win on the next turn by choosing $k = 6$.

,

B. Missing Subsequence Sum

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given two integers n and k. Find a sequence a of non-negative integers of size at most 25 such that the following conditions hold.

- There is no subsequence of a with a sum of k.
- For all $1 \leq v \leq n$ where $v \neq k$, there is a subsequence of a with a sum of v.

A sequence b is a subsequence of a if b can be obtained from a by the deletion of several (possibly, zero or all) elements, without changing the order of the remaining elements. For example, $[5, 2, 3]$ is a subsequence of $[1, 5, 7, 8, 2, 4, 3]$.

It can be shown that under the given constraints, a solution always exists.

Input

The first line of the input contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases. The description of the test cases follows.

Each test case consists of a single line containing two integers n and k ($2 \leq n \leq 10^6$, $1 \leq k \leq n$) — the parameters described above.

It is guaranteed that the sum of n over all test cases does not exceed 10^7.

Output

The first line of output for each test case should contain a single integer m ($1 \leq m \leq 25$) — the size of your chosen sequence.

The second line of output for each test case should contain m integers a_i ($0 \leq a_i \leq 10^9$) — the elements of your chosen sequence.

If there are multiple solutions, print any.

Standard Input	Standard Output
5 2 2 6 1 8 8 9 3 10 7	1 1 5 2 3 4 5 6 7 1 1 1 1 1 1 1 4 7 1 4 1 4 1 2 8 3

Note

In the first example, we just need a subsequence that adds up to 1, but not one that adds up to 2. So the array $a = [1]$ suffices.

In the second example, all elements are greater than $k = 1$, so no subsequence adds up to 1. Every other integer between 1 and n is present in the array, so there is a subsequence of size 1 adding up to each of those numbers.

,

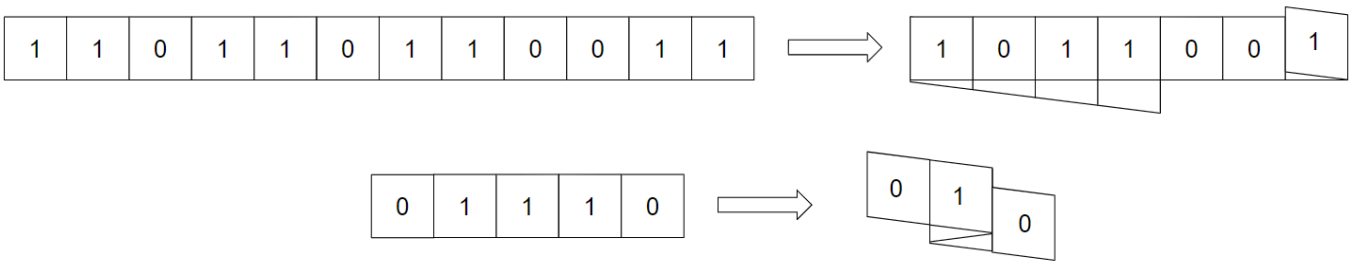
C. Folding Strip

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

You have a strip of paper with a binary string s of length n. You can fold the paper in between any pair of adjacent digits.

A set of folds is considered *valid* if after the folds, all characters that are on top of or below each other match. Note that all folds are made at the same time, so the characters don't have to match in between folds.

For example, these are valid foldings of $s = \texttt{110110110011}$ and $s = \texttt{01110}$:



The length of the folded strip is the length seen from above after all folds are made. So for the two above examples, after the folds shown above, the lengths would be 7 and 3, respectively.

Notice that for the above folding of $s = \texttt{01110}$, if we made either of the two folds on their own, that would not be a valid folding. However, because we don't check for validity until all folds are made, this folding is valid.

After performing a set of valid folds, what is the minimum length strip you can form?

Input

The first line of the input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the size of the strip.

The second line of each test case contains a string s of n characters '$\texttt{0}$' and '$\texttt{1}$' — a description of the digits on the strip.

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

Output

For each test case, output a single integer — the minimum possible length of the strip after a valid folding.

Standard Input	Standard Output
6 6 101101 1 0 12 110110110011 5	3 1 3 3 1 2

01110 4 1111 2 01	

Note

For the first example case, one optimal folding is to fold the strip in the middle, which produces a strip of length 3.

The third and fourth example cases correspond to the images above. Note that the folding shown above for $s = \texttt{110110110011}$ is not of minimal length.

,

D. Missing Subarray Sum

Input file: standard input
Output file: standard output
Time limit: 5 seconds
Memory limit: 256 megabytes

There is a hidden array a of n positive integers. You know that a is a **palindrome**, or in other words, for all $1 \le i \le n$, $a_i = a_{n+1-i}$. You are given the sums of all but one of its distinct subarrays, in arbitrary order. The subarray whose sum is not given can be any of the $\frac{n(n+1)}{2}$ distinct subarrays of a.

Recover any possible palindrome a. The input is chosen such that there is always at least one array a that satisfies the conditions.

An array b is a subarray of a if b can be obtained from a by the deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Input

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

The first line of each test case contains a single integer n ($3 \le n \le 1000$) — the size of the array a.

The next line of each test case contains $\frac{n(n+1)}{2} - 1$ integers s_i ($1 \le s_i \le 10^9$) — all but one of the subarray sums of a.

It is guaranteed that the sum of n over all test cases does not exceed 1000.

Additional constraint on the input: There is always at least one valid solution.

Hacks are disabled for this problem.

Output

For each test case, print one line containing n positive integers $a_1, a_2, \cdots a_n$ — any valid array a. Note that a must be a palindrome.

If there are multiple solutions, print any.

Standard Input	Standard Output
7 3 1 2 3 4 1 4 18 2 11 9 7 11 7 2 9 4 5 10 5 16 3 3 13 8 8 4 8 10 4 6 4 20 14 14 6 5 1 2 3 4 5 4 3 2 1 1 2 3 2 1 5 1 1 2 2 2 3 3 3 3 4 5 5 6 8	1 2 1 7 2 2 7 3 5 5 3 6 4 4 6 1 1 1 1 1 2 1 2 1 2 500000000 500000000 500000000

3 500000000 1000000000 500000000 500000000 1000000000	

Note

For the first example case, the subarrays of $a = [1, 2, 1]$ are:

- $[1]$ with sum 1,
- $[2]$ with sum 2,
- $[1]$ with sum 1,
- $[1, 2]$ with sum 3,
- $[2, 1]$ with sum 3,
- $[1, 2, 1]$ with sum 4.

So the full list of subarray sums is $1, 1, 2, 3, 3, 4$, and the sum that is missing from the input list is 3.
For the second example case, the missing subarray sum is 4, for the subarray $[2, 2]$.

For the third example case, the missing subarray sum is 13, because there are two subarrays with sum 13 ($[3, 5, 5]$ and $[5, 5, 3]$) but 13 only occurs once in the input.

,

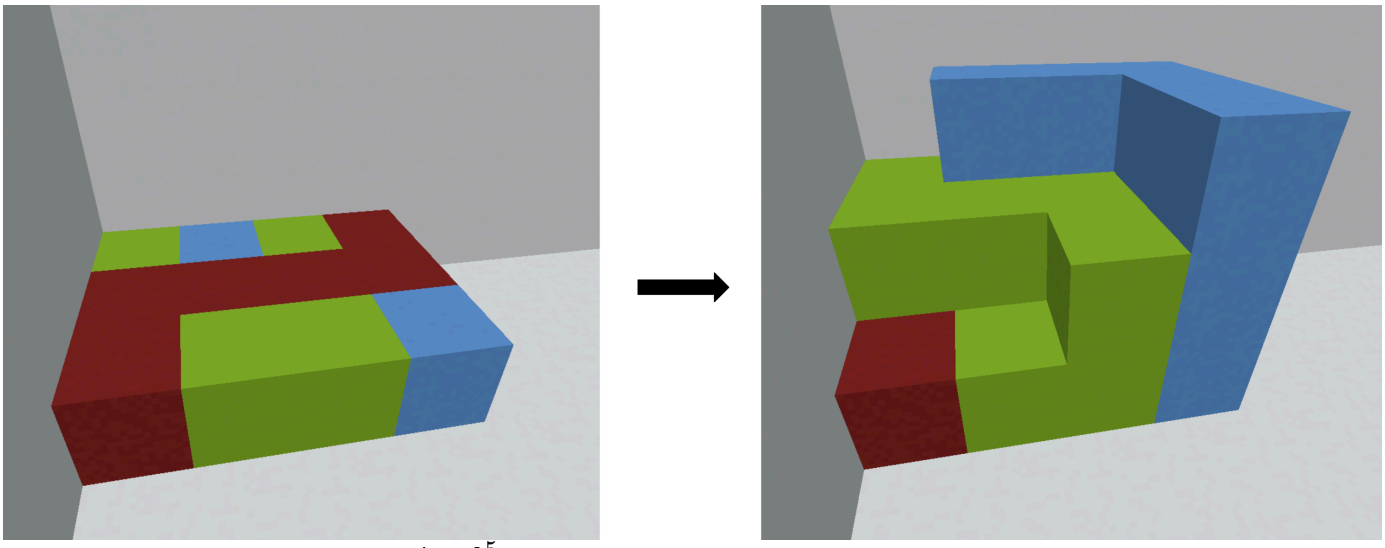
E. Connected Cubes

Input file:	standard input
Output file:	standard output
Time limit:	4 seconds
Memory limit:	256 megabytes

There are $n \cdot m$ unit cubes currently in positions $(1, 1, 1)$ through $(n, m, 1)$. Each of these cubes is one of k colors. You want to add additional cubes at any integer coordinates such that the subset of cubes of each color is connected, where two cubes are considered connected if they share a face.

In other words, for every pair of cubes of the same color c, it should be possible to travel from one to the other, moving only through cubes of color c that share a face.

The existing cubes are currently in the corner of a room. There are colorless cubes completely filling the planes $x = 0$, $y = 0$, and $z = 0$, preventing you from placing additional cubes there or at any negative coordinates.



Find a solution that uses at most $4 \cdot 10^5$ additional cubes (not including the cubes that are currently present), or determine that there is no solution. It can be shown that under the given constraints, if there is a solution, there is one using at most $4 \cdot 10^5$ additional cubes.

Input

The first line of the input contains three integers n, m, and k ($2 \le n, m, k \le 50$) — the number of rows and columns of cubes, and the number of colors, respectively.

The i-th of the next n lines contains m integers. The j-th of these is a_{ij} ($1 \le a_{ij} \le k$) — the color of the cube at position $(i, j, 1)$. For every color from 1 to k, it is guaranteed that there is at least one cube in the input of that color.

Output

If there is no solution, print a single integer -1.

Otherwise, the first line of output should contain a single integer p ($0 \le p \le 4 \cdot 10^5$) — the number of additional cubes you will add.

The next p lines should contain four integers x, y, z and c ($1 \le x, y, z \le 10^6$, $1 \le c \le k$) — indicating that you are adding a cube with color c at position (x, y, z).

No two cubes in the output should have the same coordinates, and no cube in the output should have the same coordinates as any cube in the input.

If there are multiple solutions, print any.

Standard Input	Standard Output
3 4 3 3 2 3 1 1 1 1 1 1 3 3 2	13 1 1 2 3 1 3 2 3 2 1 2 3 2 2 2 3 2 3 2 3 3 3 2 3 1 2 2 2 1 2 3 2 1 3 3 2 1 4 3 2 2 4 3 2 3 4 3 2 3 4 2 2
2 2 2 2 1 1 2	9 1 3 1 1 2 3 1 1 3 1 1 1 3 2 1 1 3 3 1 1 1 1 2 2 1 2 2 2 2 1 2 2 2 2 2 2

Note

The image in the statement corresponds to the first example case, with $\mathrm{red} = 1$, $\mathrm{blue} = 2$, $\mathrm{green} = 3$.

,

F. Conference

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 512 megabytes

You have been asked to organize a very important art conference. The first step is to choose the dates.

The conference must last for a certain number of consecutive days. Each day, one lecturer must perform, and the same lecturer cannot perform more than once.

You asked n potential lecturers if they could participate in the conference. Lecturer i indicated that they could perform on any day from l_i to r_i inclusive.

A certain segment of days can be chosen as the conference dates if there is a way to assign an available lecturer to each day of the segment, assigning each lecturer to no more than one day.

For each k from 1 to n, find how many ways there are to choose a segment of k consecutive days as the conference dates.

Input

The first line of input contains one integer n — the number of potential lecturers ($1 \le n \le 2 \cdot 10^5$).

Each of the next n lines contains two integers l_i and r_i — the segment of available days for the ith lecturer ($1 \le l_i \le r_i \le 2 \cdot 10^5$).

Output

Print n integers, where the kth number denotes the number of ways to select a segment of k consecutive days as conference dates.

Standard Input	Standard Output
3 1 2 3 4 5 6	6 2 0
5 1 3 1 3 1 3 1 3 1 3	3 2 1 0 0

Note

In the first testcase, a one-day conference can be organized on any of the days from 1 to 6. A two-day conference can be organized from day 2 to day 3, as well as from day 4 to day 5.

In the second testcase, five lecturers can perform only from day 1 to day 3, so it will not be possible to organize a conference longer than three days.