VK Cup 2021 - Elimination (Engine)

A. Binary Decimal

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

Let's call a number a *binary decimal* if it's a positive integer and all digits in its decimal notation are either 0 or 1. For example, $1\,010\,111$ is a binary decimal, while $10\,201$ and $787\,788$ are not.

Given a number n, you are asked to represent n as a sum of some (not necessarily distinct) binary decimals. Compute the smallest number of binary decimals required for that.

Input

The first line contains a single integer t ($1 \le t \le 1000$), denoting the number of test cases.

The only line of each test case contains a single integer n ($1 \le n \le 10^9$), denoting the number to be represented.

Output

For each test case, output the smallest number of binary decimals required to represent n as a sum.

Example

input
3 121 5 1000000000
output
2 5 1

Note

In the first test case, 121 can be represented as $121 = 110 + 11$ or $121 = 111 + 10$.

In the second test case, 5 can be represented as $5 = 1 + 1 + 1 + 1 + 1$.

In the third test case, $1\,000\,000\,000$ is a binary decimal itself, thus the answer is 1.

B. Putting Plates

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

To celebrate your birthday you have prepared a festive table! Now you want to seat as many guests as possible.

The table can be represented as a rectangle with height h and width w, divided into $h \times w$ cells. Let (i, j) denote the cell in the i-th row and the j-th column of the rectangle ($1 \le i \le h$; $1 \le j \le w$).

Into each cell of the table you can either put a plate or keep it empty.

As each guest has to be seated next to their plate, you can only put plates on the edge of the table — into the first or the last row of the rectangle, or into the first or the last column. Formally, for each cell (i, j) you put a plate into, at least one of the following conditions must be satisfied: $i = 1$, $i = h$, $j = 1$, $j = w$.

To make the guests comfortable, no two plates must be put into cells that have a common side or corner. In other words, if cell (i, j) contains a plate, you can't put plates into cells $(i - 1, j)$, $(i, j - 1)$, $(i + 1, j)$, $(i, j + 1)$, $(i - 1, j - 1)$, $(i - 1, j + 1)$, $(i + 1, j - 1)$, $(i + 1, j + 1)$.

Put as many plates on the table as possible without violating the rules above.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases.

Each of the following t lines describes one test case and contains two integers h and w ($3 \le h, w \le 20$) — the height and the width of the table.

Output
For each test case, print h lines containing w characters each. Character j in line i must be equal to 1 if you are putting a plate into cell (i, j), and 0 otherwise. If there are multiple answers, print any.

All plates must be put on the edge of the table. No two plates can be put into cells that have a common side or corner. The number of plates put on the table under these conditions must be as large as possible.

You are allowed to print additional empty lines.

Example

input
3 3 5 4 4 5 6
output
10101 00000 10101 0100 0001 1000 0010 010101 000000 100001 000000 101010

Note
For the first test case, example output contains the only way to put 6 plates on the table.

For the second test case, there are many ways to put 4 plates on the table, example output contains one of them.

Putting more than 6 plates in the first test case or more than 4 plates in the second test case is impossible.

C. Pursuit

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

You and your friend Ilya are participating in an individual programming contest consisting of multiple stages. A contestant can get between 0 and 100 points, inclusive, for each stage, independently of other contestants.

Points received by contestants in different stages are used for forming overall contest results. Suppose that k stages of the contest are completed. For each contestant, $k - \lfloor \frac{k}{4} \rfloor$ stages with the highest scores are selected, and these scores are added up. This sum is the overall result of the contestant. (Here $\lfloor t \rfloor$ denotes rounding t down.)

For example, suppose 9 stages are completed, and your scores are $50, 30, 50, 50, 100, 10, 30, 100, 50$. First, 7 stages with the highest scores are chosen — for example, all stages except for the 2-nd and the 6-th can be chosen. Then your overall result is equal to $50 + 50 + 50 + 100 + 30 + 100 + 50 = 430$.

As of now, n stages are completed, and you know the points you and Ilya got for these stages. However, it is unknown how many more stages will be held. You wonder what the smallest number of additional stages is, after which your result might become greater than or equal to Ilya's result, at least in theory. Find this number!

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the number of completed stages.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 100$) — your points for the completed stages.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 100$) — Ilya's points for the completed stages.

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

Output
For each test case print a single integer — the smallest number of additional stages required for your result to be able to become greater than or equal to Ilya's result.

If your result is already not less than Ilya's result, print 0.

Example

input
5 1 100 0 1 0 100 4 20 30 40 50 100 100 100 100 4 10 20 30 40 100 100 100 100 7 7 59 62 52 27 31 55 33 35 50 98 83 80 64

output
0 1 3 4 2

Note

In the first test case, you have scored 100 points for the first stage, while Ilya has scored 0. Thus, your overall result (100) is already not less than Ilya's result (0).

In the second test case, you have scored 0 points for the first stage, while Ilya has scored 100. A single stage with an opposite result is enough for both your and Ilya's overall scores to become equal to 100.

In the third test case, your overall result is $30 + 40 + 50 = 120$, while Ilya's result is $100 + 100 + 100 = 300$. After three additional stages your result might become equal to 420, while Ilya's result might become equal to 400.

In the fourth test case, your overall result after four additional stages might become equal to 470, while Ilya's result might become equal to 400. Three stages are not enough.

D. Secret Santa

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

Every December, VK traditionally holds an event for its employees named "Secret Santa". Here's how it happens.

n employees numbered from 1 to n take part in the event. Each employee i is assigned a different employee b_i, to which employee i has to make a new year gift. Each employee is assigned to exactly one other employee, and nobody is assigned to themselves (but two employees may be assigned to each other). Formally, all b_i must be distinct integers between 1 and n, and for any i, $b_i \neq i$ must hold.

The assignment is usually generated randomly. This year, as an experiment, all event participants have been asked who they wish to make a gift to. Each employee i has said that they wish to make a gift to employee a_i.

Find a valid assignment b that maximizes the number of fulfilled wishes of the employees.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^5$). Description of the test cases follows.

Each test case consists of two lines. The first line contains a single integer n ($2 \leq n \leq 2 \cdot 10^5$) — the number of participants of the event.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$; $a_i \neq i$) — wishes of the employees in order from 1 to n.

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 two lines.

In the first line, print a single integer k ($0 \leq k \leq n$) — the number of fulfilled wishes in your assignment.

In the second line, print n distinct integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq n$; $b_i \neq i$) — the numbers of employees assigned to employees $1, 2, \ldots, n$.

k must be equal to the number of values of i such that $a_i = b_i$, and must be as large as possible. If there are multiple answers, print any.

Example

input
2 3 2 1 2 7 6 4 6 2 4 5 6

output
2 3 1 2 4 6 4 7 2 3 5 1

Note
In the first test case, two valid assignments exist: $[3, 1, 2]$ and $[2, 3, 1]$. The former assignment fulfills two wishes, while the latter assignment fulfills only one. Therefore, $k = 2$, and the only correct answer is $[3, 1, 2]$.

E. Minimax

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

Prefix function of string $t = t_1t_2 \ldots t_n$ and position i in it is defined as the length k of the longest proper (not equal to the whole substring) prefix of substring $t_1t_2 \ldots t_i$ which is also a suffix of the same substring.

For example, for string $t =$ abacaba the values of the prefix function in positions $1, 2, \ldots, 7$ are equal to $[0, 0, 1, 0, 1, 2, 3]$.

Let $f(t)$ be equal to the *maximum* value of the prefix function of string t over all its positions. For example, $f($abacaba$) = 3$.

You are given a string s. Reorder its characters arbitrarily to get a string t (the number of occurrences of any character in strings s and t must be equal). The value of $f(t)$ must be *minimized*. Out of all options to minimize $f(t)$, choose the one where string t is the *lexicographically smallest*.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). Description of the test cases follows.

The only line of each test case contains string s ($1 \le |s| \le 10^5$) consisting of lowercase English letters.

It is guaranteed that the sum of lengths of s over all test cases does not exceed 10^5.

Output
For each test case print a single string t.

The multisets of letters in strings s and t must be equal. The value of $f(t)$, the maximum of prefix functions in string t, must be as small as possible. String t must be the lexicographically smallest string out of all strings satisfying the previous conditions.

Example

input
3 vkcup abababa zzzzzz

output
ckpuv aababab zzzzzz

Note
A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

In the first test case, $f(t) = 0$ and the values of prefix function are $[0, 0, 0, 0, 0]$ for any permutation of letters. String ckpuv is the lexicographically smallest permutation of letters of string vkcup.

In the second test case, $f(t) = 1$ and the values of prefix function are $[0, 1, 0, 1, 0, 1, 0]$.

In the third test case, $f(t) = 5$ and the values of prefix function are $[0, 1, 2, 3, 4, 5]$.

F. Bingo

time limit per test: 7 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Getting ready for VK Fest 2021, you prepared a table with n rows and n columns, and filled each cell of this table with some event related with the festival that could either happen or not: for example, whether you will win a prize on the festival, or whether it will rain.

Forecasting algorithms used in VK have already estimated the probability for each event to happen. Event in row i and column j will happen with probability $a_{i,j} \cdot 10^{-4}$. All of the events are mutually independent.

Let's call the table *winning* if there exists a line such that all n events on it happen. The line could be any horizontal line (cells $(i, 1), (i, 2), \ldots, (i, n)$ for some i), any vertical line (cells $(1, j), (2, j), \ldots, (n, j)$ for some j), the main diagonal (cells $(1, 1), (2, 2), \ldots, (n, n)$), or the antidiagonal (cells $(1, n), (2, n - 1), \ldots, (n, 1)$).

Find the probability of your table to be winning, and output it modulo $31\,607$ (see Output section).

Input

The first line contains a single integer n ($2 \le n \le 21$) — the dimensions of the table.

The i-th of the next n lines contains n integers $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ ($0 < a_{i,j} < 10^4$). The probability of event in cell (i, j) to happen is $a_{i,j} \cdot 10^{-4}$.

Output

Print the probability that your table will be winning, modulo $31\,607$.

Formally, let $M = 31\,607$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \bmod M$. In other words, output such an integer x that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Examples

input
2 5000 5000 5000 5000
output
5927

input
2 2500 6000 3000 4000
output
24812

input
3 1000 2000 3000 4000 5000 6000 7000 8000 9000
output
25267

Note

In the first example, any two events form a line, and the table will be winning if any two events happen. The probability of this is $\frac{11}{16}$, and $5927 \cdot 16 \equiv 11 \pmod{31\,607}$.

G. What a Reversal

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

You have two strings a and b of equal length n consisting of characters `0` and `1`, and an integer k.

You need to make strings a and b equal.

In one step, you can choose any substring of a containing exactly k characters `1` (and arbitrary number of characters `0`) and reverse it. Formally, if $a = a_1 a_2 \ldots a_n$, you can choose any integers l and r ($1 \le l \le r \le n$) such that there are exactly k ones among characters $a_l, a_{l+1}, \ldots, a_r$, and set a to $a_1 a_2 \ldots a_{l-1} a_r a_{r-1} \ldots a_l a_{r+1} a_{r+2} \ldots a_n$.

Find a way to make a equal to b using at most $4n$ reversals of the above kind, or determine that such a way doesn't exist. The number of reversals doesn't have to be minimized.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 2000$). Description of the test cases follows.

Each test case consists of three lines. The first line of each test case contains two integers n and k ($1 \le n \le 2000$; $0 \le k \le n$).

The second line contains string a of length n.

The third line contains string b of the same length. Both strings consist of characters `0` and `1`.

It is guaranteed that the sum of n over all test cases does not exceed 2000.

Output
For each test case, if it's impossible to make a equal to b in at most $4n$ reversals, print a single integer -1.

Otherwise, print an integer m ($0 \le m \le 4n$), denoting the number of reversals in your sequence of steps, followed by m pairs of integers l_i, r_i ($1 \le l_i \le r_i \le n$), denoting the boundaries of the substrings of a to be reversed, in chronological order. Each substring must contain exactly k ones at the moment of reversal.

Note that m doesn't have to be minimized. If there are multiple answers, print any.

Example

input
6 6 1 101010 010101 6 3 101010 010101 6 0 101010 010101 6 6 101010 010101 4 2 0000 1111 9 2 011100101 101001011
output
3 1 2 3 4 5 6 1 1 6 -1 -1 -1 5 4 8 8 9 3 6 1 4 3 6

Note
In the first test case, after the first reversal $a =$ `011010`, after the second reversal $a =$ `010110`, after the third reversal $a =$ `010101`.

H. Turing's Award

time limit per test: 10 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Alan Turing is standing on a tape divided into cells that is infinite in both directions.

Cells are numbered with consecutive integers from left to right. Alan is initially standing in cell 0. Every cell x has cell $x - 1$ on the left and cell $x + 1$ on the right.

Each cell can either contain an integer or be empty. Initially all cells are empty.

Alan is given a permutation $a_1, a_2, \ldots, a_n$ of integers from 1 to n that was chosen **uniformly at random** among all permutations of length n.

At time 1, integer a_1 is written down into cell 0 where Alan is located.

At each time i from 2 to n inclusive, the following happens. First, Alan decides whether to stay in the same cell he's currently in, move to the neighboring cell on the left, or move to the neighboring cell on the right. After that, integer a_i is written down into the cell where Alan is located. If that cell already contained some integer, the old integer is overwritten and irrelevant from that moment on.

Once a_n is written down into some cell at time n, sequence b of all integers contained in the cells from left to right is formed. Empty cells are ignored.

Turing's award is equal to the length of the longest increasing subsequence of sequence b.

Help Alan and determine the largest possible value of his award if he acts optimally.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). Description of the test cases follows.

Each test case is given in two lines. The first line of each test case contains a single integer n ($2 \le n \le 15\,000$) — the length of the permutation given to Alan.

The second line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the permutation.

It is guaranteed that the permutation was chosen uniformly at random among all permutations of the corresponding length.

The sum of n over all test cases does not exceed $15\,000$.

Output

For each test case output a single integer — the largest possible value of Turing's award.

Hacks are not allowed for this problem.

Example

input
4 2 1 2 4 4 1 2 3 7 3 6 5 7 4 1 2 7 5 2 3 7 6 1 4
output
2 3 4 4

Note

Longest increasing subsequence of sequence b is the longest increasing sequence that can be obtained from b by deletion of several (possibly, zero or all) elements.

In the first test case, Alan can make a decision only at time 2. If Alan stays in cell 0, sequence b will be equal to $[2]$. If Alan moves to the left, to cell -1, sequence b will be equal to $[2, 1]$. If Alan moves to the right, to cell 1, sequence b will be equal to $[1, 2]$. Only in the last case the length of the longest increasing subsequence of b is 2, therefore, the answer is equal to 2.

In the second test case, one of the optimal sequences of actions looks as follows: move to the left at times 2 and 3, and move to the right at time 4. Then sequence b will be equal to $[2, 3, 4]$, and the length of its longest increasing subsequence is 3.

In the third test case, one of the best ways is to always move to the left. Then sequence b will be equal to $[2, 1, 4, 7, 5, 6, 3]$, and the length of its longest increasing subsequence is 4.

In the fourth test case, one of the best ways is to move to the right four times, then move to the left once, and stay in the same cell once. Sequence b will be equal to $[5, 2, 3, 4, 6]$.