

Problem A. Stones and Bananas

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

A boy named Petya has a grid of paper sized $n \times n$, with rows and columns numbered from 1 to n , left to right and top to bottom, respectively. Thus, each cell is identified by a pair of numbers (x, y) , where x indicates the row number and y indicates the column number.

Petya placed two bananas in cells (x_1, y_1) and (x_2, y_2) , after which some cells on the paper magically became occupied by hot stones. Petya can move no more than k stones before they burn through the paper.

Each stone burns the cell of the paper where it is located **after the move**, which can cause the paper to break into several pieces. Petya does not want the bananas to be sad, so they must ultimately end up on the same piece of paper. Help Petya move the stones so that the bananas are not sad, or tell him that it is impossible.

Input

The first line contains two integers n and k ($1 \leq n \leq 1000$, $0 \leq k \leq 5$).

The second line contains two numbers x_1 and y_1 ($1 \leq x_1, y_1 \leq n$) — the coordinates of the first banana.

The third line contains two numbers x_2 and y_2 ($1 \leq x_2, y_2 \leq n$) — the coordinates of the second banana.

The following n lines contain binary strings of length n , where the j -th character of the i -th string is 1 if there is a stone in cell (i, j) , and 0 otherwise.

It is guaranteed that there are no stones in cells (x_1, y_1) and (x_2, y_2) .

Output

In the first line, output «Yes» if Petya can ensure that the bananas are not sad, and «No» otherwise.

If you output «Yes», then in the second line output the number m ($0 \leq m \leq k$) — the number of stones that Petya needs to move.

Then output m lines, each containing four numbers f_x, f_y, p_x, p_y ($1 \leq f_x, f_y, p_x, p_y \leq n$), indicating that Petya should move a stone from cell (f_x, f_y) to cell (p_x, p_y) .

Note that you cannot move the same stone twice, nor can you move a stone to a cell with a banana or to a cell that already contains another stone or has contained one previously.

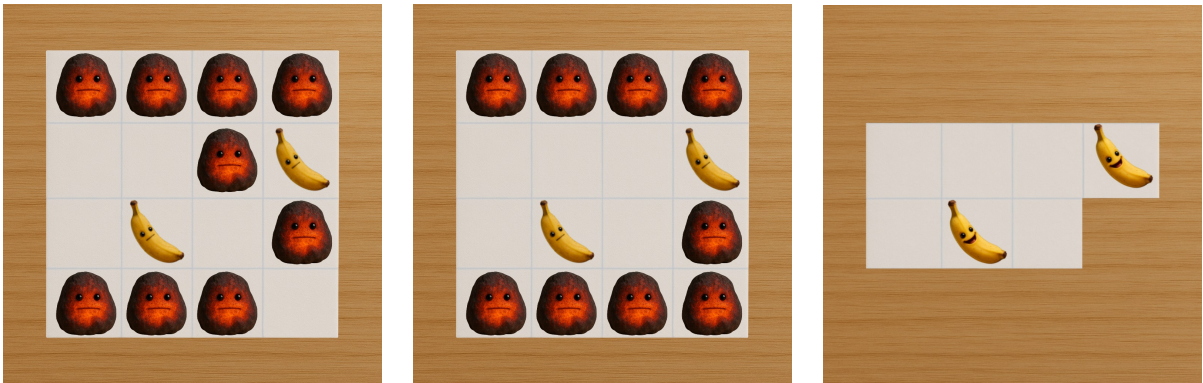
If there are multiple valid answers, you may output any of them.

Examples

standard input	standard output
4 1 3 1 2 4 1111 0010 0001 1110	Yes 1 2 3 4 4
2 0 1 1 2 2 01 10	No

Note

In the first example, Petya should move the stone from position (2,3) to position (4,4).



The images show: the initial arrangement of the bananas and stones, the arrangement after the move, and the surviving piece of paper with two happy bananas.

In the second example, Petya cannot move the stones, so the paper will break into two pieces, each containing one banana. The cells (1,1) and (2,2) do not form a connected figure, as they are adjacent diagonally, not by sides.

Problem B. Code Lock

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 512 megabytes

Ulyana is very concerned about the safety of her belongings, so she uses code locks.

A code lock consists of a set of n rotors, on the i -th of which are written numbers from 0 to $m_i - 1$ inclusive. The state of the lock is determined by an array of numbers $a_1, \dots, a_n$ ($0 \leq a_i < m_i$) — the numbers set on each of the rotors. Initially, all a_i are equal to 0.

Ulyana has set the combination $b_1, \dots, b_n$ ($0 \leq b_i \leq m_i - 1$) as the lock code. The lock will open if, as a result of manipulations with the lock, it turns out that $a_i = b_i$ for all $i \in \{1, 2, \dots, n\}$.

In one action, Ulyana can choose a segment $[l, r]$ ($1 \leq l \leq r \leq n$) and *rotate* all the rotors in this segment, after which for each i ($l \leq i \leq r$) the value a_i will be replaced with $(a_i + 1) \bmod m_i$.

Ulyana wants to know how quickly she can open her lock, if necessary. Help her determine the minimum number of actions required to do this.

Input

Each test consists of several sets of input data. The first line contains a single integer t ($1 \leq t \leq 500$) — the number of sets of input data. The descriptions of the sets follow.

In the first line of each set, there is a number n ($1 \leq n \leq 500$) — the number of rotors in the lock.

In the second line of each set, there are numbers $m_1, m_2, \dots, m_n$ ($2 \leq m_i \leq 500$) — the number of possible values on each of the rotors.

In the third line of each set, there are numbers $b_1, b_2, \dots, b_n$ ($0 \leq b_i \leq m_i - 1$) — the combination of values that unlocks the lock.

It is guaranteed that the sum of n across all sets of input data does not exceed 500.

Output

For each lock, output on a separate line the minimum number of actions required to open it.

Example

standard input	standard output
3	4
4	10
5 5 5 5	3
1 2 3 4	
5	
10 2 10 2 10	
8 1 8 1 8	
7	
4 4 3 2 3 4 4	
1 3 0 1 0 3 1	

Note

Let us denote the action on the segment from l to r as $[l, r]$.

The first lock can be opened in 4 consecutive actions: $[1, 4]$, $[2, 4]$, $[3, 4]$, $[4, 4]$.

The second lock can be opened by performing 8 actions $[1, 5]$, 1 action $[2, 2]$, and 1 action $[4, 4]$.

The third lock can be opened by performing 3 consecutive actions: $[1, 7]$, $[2, 6]$, $[2, 6]$.

Problem C. Good Colorings – 7

Input file: `standard input`
Output file: `standard output`
Time limit: 1 second
Memory limit: 512 megabytes

The cells of an $n \times n$ table are colored in three colors: red, blue, and white. It is required to repaint the white cells in either blue or red (different cells can be repainted in different colors) such that the following conditions are satisfied:

- In any row, there is an **even** number of red cells;
- In any column, there is an **even** number of red cells;
- On the main diagonal of the table, there is an **even** number of red cells;
- On the secondary diagonal of the table, there is an **even** number of red cells.

The main and secondary diagonals of a square $n \times n$ consist of a pair of diagonals, each containing exactly n cells.

Input

Each test consists of several sets of input data. The first line contains one integer t ($1 \leq t \leq 2\,000$) — the number of input data sets. The description of the input data sets follows.

The first line of each input data set contains the number n ($1 \leq n \leq 1\,000$) — the size of the table.

The next n lines contain n consecutive characters "R" "B" "W"— the colors of the cells in the table:

- The character "R"denotes that the corresponding cell is red;
- The character "B"denotes that the corresponding cell is blue;
- The character "W"denotes that the corresponding cell is white.

It is guaranteed that the sum of n across all sets does not exceed 10^4 .

Output

Print "Yes"(without quotes) if it is possible to repaint the white cells of the table such that the conditions of the problem are satisfied, and "No"otherwise.

In the case of a positive answer to the problem, print n lines of n characters "R"and "B—the desired coloring of the table.

Example

standard input	standard output
3	Yes
3	RBR
RBR	BBB
WWW	RBR
WWR	No
2	Yes
BW	RRBB
WR	BRBR
4	RBRB
WRWB	BBRR
WWW	
RWWW	
BBWW	

Problem D. How to Annoy a Turtle

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 256 megabytes

Given a table of size $h \times w$, where the rows are numbered from top to bottom with numbers from 1 to h , and the columns are numbered from left to right with numbers from 1 to w . We denote the cell at the intersection of the i -th row and the j -th column as the pair (i, j) .

The turtle lives in cell $(1, 1)$ and wants to reach cell (h, w) , where her friend lives. As is known, turtles living in the table can only move left-to-right and top-to-bottom. Specifically, from cell (i, j) , the turtle can move to cells $(i + 1, j)$ and $(i, j + 1)$, provided these cells exist.

Evil Sasha decided to block some cells in the table so that the turtle **cannot** reach her friend. As one might guess, the turtle will not be able to visit blocked cells. However, Sasha does not want to appear as a villain, so he will not block any two adjacent cells either by side or corner, nor will he block cells $(1, 1)$ and (h, w) .

Check if Sasha can block some cells so that the turtle cannot reach cell $(1, 1)$ to cell (h, w) .

Input

The input consists of a single line containing a pair of numbers h and w ($2 \leq h, w \leq 200$).

Output

Print «No» if Sasha cannot block the turtle's path.

Otherwise, print «Yes» on the first line, and in the following h lines, print w characters «X» or «.» — the table with marked blocked cells. The character «.» denotes a regular cell, while the character «X» — denotes a blocked cell.

Examples

standard input	standard output
2 2	No
3 7	Yes .X...X. ...X... .X...X.

Problem E. Fight Club

Input file: **standard input**
Output file: **standard output**
Time limit: 2 seconds
Memory limit: 512 megabytes

As is well known, gorillas are strong and intelligent animals. After scientists taught one of the gorillas the Zig programming language, she became enlightened, broke free, and united all the gorillas of the world into a society of the future. Clearly, her knowledge must be passed on to future generations, so the gorilla founded a university with a single faculty where she taught the Zig language and algorithms in distributed systems.

A total of n gorillas wanted to enroll in the university, with their strengths being $a_1, \dots, a_n$. It is assumed that no two gorillas have the same strength, so a_i are distinct numbers from 1 to n .

The enrollment is organized as follows. First, an arbitrary segment $[l, r]$ is chosen ($l < r$), where it is announced that the two strongest gorillas will enroll in the university. Gorillas are not only strong and intelligent animals but also very determined, so if at least one of the two strongest gorillas is not at the edge of the segment, she is immediately rejected and disqualified from the competition.

The founding gorilla understands this, so she tries to choose only *honest* segments for the competition, meaning those where the two largest values are located at the boundaries of the segment. In other words, the segment $[l, r]$ is *honest* if $\min\{a_l, a_r\} > a_i$ for all $l < i < r$. In particular, a segment consisting of two gorillas is always *honest*.

But while the founding gorilla is thinking about how to organize the competition, the other gorillas are continuously training. As a result, in the following q hours, the following occurs: in the j -th hour ($1 \leq j \leq q$), the gorilla with number x_j goes to the gym and becomes the new strongest gorilla, after which, in this and all subsequent hours (until she goes to the gym again), her strength is considered to be $n + j$.

Now the founding gorilla is completely confused and wants to at least understand how many ways she will have to conduct the competition *honestly* at each moment in time.

In other words, for each moment in time j ($1 \leq j \leq q$), it is required to calculate the number of *honest* segments at that moment in time.

Input

The first line contains two positive integers n and q ($1 \leq n, q \leq 10^6$) — the number of gorilla applicants and the number of hours.

The second line contains n positive integers $a_1, \dots, a_n$ ($1 \leq a_i \leq n$) — the strengths of the gorillas at the initial moment in time. It is guaranteed that all numbers a_i are distinct.

The third line contains q numbers $x_1, \dots, x_q$ ($1 \leq x_i \leq n$), where x_j — the number of the gorilla that goes to the gym in the j -th hour.

Output

In a single line, output q numbers, the i -th of which is equal to the number of *honest* segments at moment in time i .

Example

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

Problem F. Arsen and Toy Soldiers

Input file: **standard input**
Output file: **standard output**
Time limit: 3 seconds
Memory limit: 512 megabytes

Arsen commands a squad of n toy soldiers. Every morning and every evening, a formation is held. At each formation, each soldier is given a rifle of one of n types.

Before the start of the day, Arsen has two partially filled schedules: $a_1, \dots, a_n$ and $b_1, \dots, b_n$. Here, a_i is the type of rifle that the i -th soldier receives at the morning formation, and b_i is the type of rifle he receives at the evening formation. If a_i or b_i is equal to -1 , it means that the type of rifle for the corresponding formation has not yet been assigned. Arsen can replace each -1 with any number from 1 to n .

After the schedules for issuing rifles are fully determined, Arsen conducts another ritual: he decides whether to issue soap to each soldier or not. Formally, he creates a binary array c , where $c_i = 0$ or $c_i = 1$. This array must satisfy two balance conditions:

- For each type of rifle k (from 1 to n), among all soldiers who received a rifle of type k at the morning formation, the number of soldiers with soap ($c_i = 1$) must match the number of soldiers without soap ($c_i = 0$).
- Similarly, for each type of rifle k , among all soldiers who received a rifle of type k at the evening formation, there must also be an equal number of soldiers with soap and without.

Help Arsen determine if it is possible to fill in the schedules a and b (replacing all -1 s with numbers from 1 to n) such that there exists at least one way to issue soap (array c) that satisfies the balance conditions. If this is possible, reconstruct an example of suitable arrays a , b , and c .

Input

The first line contains an integer n ($1 \leq n \leq 10^6$) — the number of soldiers.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq n$ or $a_i = -1$) — the assignments of rifles for the morning formation.

The third line contains n integers $b_1, b_2, \dots, b_n$ ($1 \leq b_i \leq n$ or $b_i = -1$) — the assignments of rifles for the evening formation.

Output

In the first line, print "Yes" if it is possible to fill in the arrays a and b so that the pair of arrays becomes good. Otherwise, print "No".

If the answer is "Yes" print three additional lines:

- In the second line, print n integers — the complete array a after filling all -1 s.
- In the third line, print n integers — the complete array b after filling all -1 s.
- In the fourth line, print n integers (0 or 1) — the array c that satisfies the balance conditions.

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

Examples

standard input	standard output
6 -1 1 -1 2 -1 2 1 -1 2 -1 2 -1	YES 1 1 1 2 1 2 1 1 2 1 2 1 0 1 0 1 1 0
4 1 1 1 1 1 1 1 2	NO

Problem G. Geometry!

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

Anton does not like long legends, but he loves geometry, so here is the formal problem statement.

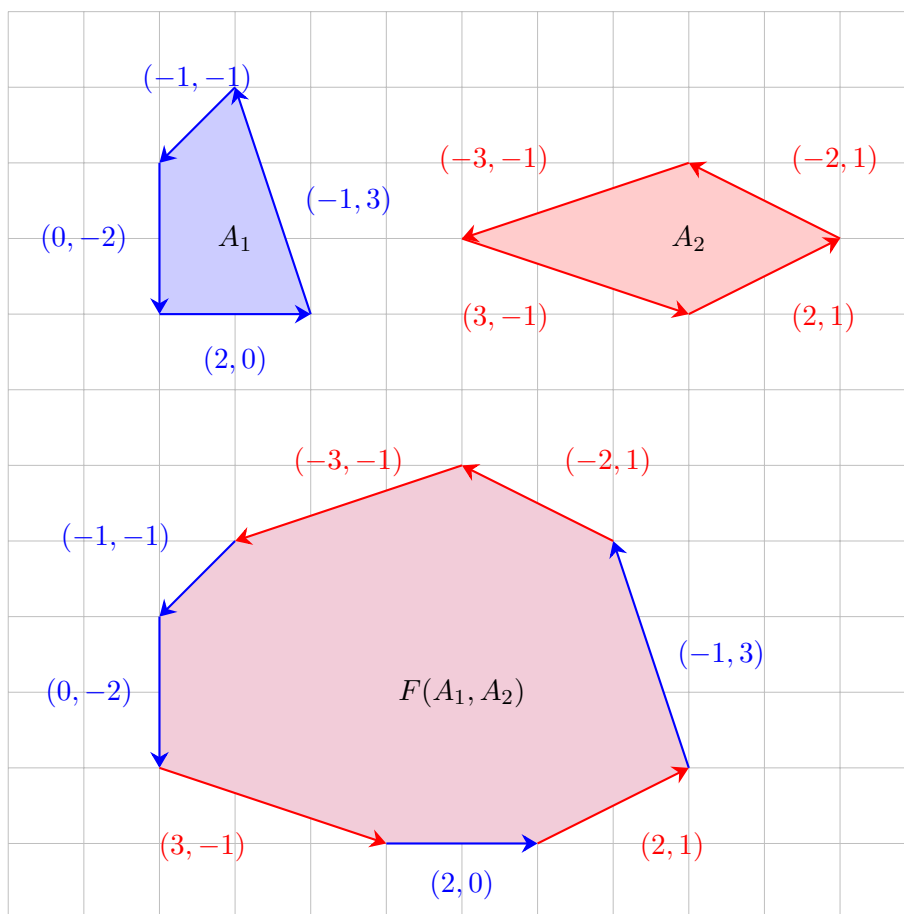
You are given n convex polygons $A_1, A_2, \dots, A_n$. Polygon number i consists of k_i vertices, and, accordingly, k_i edges. It is **guaranteed that no two edges of the polygons are parallel** (i.e., no polygon has parallel edges, and edges of different polygons are also not parallel). Each polygon is defined as a set of edges, each edge is defined as a vector, and the vectors are listed in counterclockwise order (the directions of the vectors also go counterclockwise).

We define a function $F(A, B)$, where A, B are convex polygons. $F(A, B)$ is a convex polygon formed from all edges of polygons A and B , meaning that each edge is one of the edges from A or B , while the direction and length of this edge must be preserved, and each edge must be used exactly once. It can be proven that there is always exactly one such convex polygon.

We define a function $G(A, B)$, where A, B are convex polygons. $G(A, B)$ is the number of such vertices in $F(A, B)$ for which one of the adjacent edges is taken from A and the other from B (note that since there are no parallel edges, it is always possible to determine from which polygon a given edge is taken).

You need to calculate $\sum_{i=1}^{n-1} \sum_{j=i+1}^n G(A_i, A_j)$.

For better understanding, consider the picture from the first example.



Here are the polygons A_1 and A_2 , as well as $F(A_1, A_2)$. It is easy to see that in the octagon $F(A_1, A_2)$ there are exactly 6 vertices that are adjacent to edges taken from different polygons. Thus, $G(A_1, A_2) = 6$.

Input

The first line contains an integer T — the number of independent test cases ($1 \leq T \leq 10^5$). The descriptions of T test cases follow.

The description of each test case begins with the number n on a separate line — the number of polygons ($2 \leq n \leq 10^5$).

Next are the descriptions of the polygons. For each polygon, the first line contains the number k_i — the number of vertices in the polygon. This is followed by k_i ($3 \leq k_i \leq 10^6$) lines, each describing the next edge of the polygon. Each line contains a pair of numbers x and y ($-10^9 \leq x, y \leq 10^9$) — the coordinates of the vector defining the next edge. It is guaranteed that these vectors define valid convex polygons.

It is guaranteed that the sum of n across all test cases does not exceed 10^5 . It is guaranteed that the sum of all k_i across all test cases does not exceed 10^6 .

Output

For each test case, output on a separate line one integer — $\sum_{i=1}^{n-1} \sum_{j=i+1}^n G(A_i, A_j)$.

Example

standard input	standard output
1 2 4 -1 -1 0 -2 2 0 -1 3 4 2 1 -2 1 -3 -1 3 -1	6

Problem H. Tram System

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 512 megabytes

A boy named Misha, after reading "The Master and Margarita," became afraid of trams, tracks, and everything related to them. To overcome his fear, Misha decided to build a tram system in a well-known computer game about cubes and study the properties of the constructed system.

Misha chose an even number n and built n^2 stations, forming an $n \times n$ grid, then connected them using $\frac{n}{2}$ cyclic non-intersecting railways, forming nested squares.

At each of the n^2 stations, Misha placed one tram, each of which had one of 26 types, denoted by lowercase letters of the Latin alphabet. Since the stations form an $n \times n$ grid, it is easy to write the types of all the trams in a table.

It should be clarified that Misha was not so much afraid of the trams themselves, but of the *dark forces*. Therefore, he carefully monitored the amount of *mysticism*, which can be calculated as follows:

Consider the table with the tram types at each station, then all tram types are written in a single string s (first the first row of the table is written, then the second, and so on). The *mysticism* is defined as the length of the longest prefix of s that matches the reversed suffix of s of the same length.

Misha understands that if the tram system is launched, the trams will change their positions; specifically, on each track, the trams can shift in a cycle, and the shift may occur by different amounts on different tracks. Misha is interested in the maximum possible value of *mysticism* across all possible tram positions.

Unfortunately, Misha is not as smart as you are, so he asks you to help him calculate the value he is interested in.

Input

The first line contains an **even** natural number n ($2 \leq n \leq 1000$).

The following n lines describe the configuration of the tram system—each line contains n lowercase Latin letters written consecutively—the types of trams located in that row.

Output

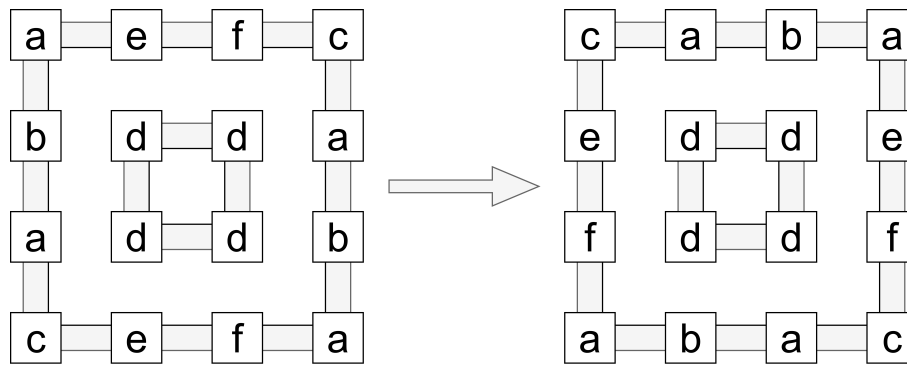
In a single line, output a single number—the maximum possible value of *mysticism*.

Examples

standard input	standard output
4 aefc bdda addb cefa	4
4 aaaa aaaa aaaa aaaa	16

Note

Consider the first example. In it, the outer tram line can be cyclically shifted by 3 trams clockwise. Then the following situation arises:



If after that all tram types are written in a single line, the resulting string will be ca**b**aed**b**efdd**f**a**b**ac. It is clear that in this string, the prefix of length 4 matches the reversed suffix of length 4, so the *mysticism* after the specified manipulations equals 4. It can be shown that a greater value of *mysticism* is impossible. In the second example, trams can be cyclically shifted arbitrarily; in any case, the resulting string will have the form **aaa** . . . **aaa**, which is itself a suffix and prefix, and is also a palindrome, so the *mysticism* equals 16.

Problem I. Rick's Grades

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

Morty has fallen behind and started getting bad grades in school. For the first time in his life, Rick didn't ignore this and decided to help. To motivate Morty, he wants to prove that he, Rick, did better in school. The school gradebook hasn't changed for many years, so both Rick and Morty received grades from 1 to 5, or 0 — which is analogous to the Russian "H" meaning that the student was absent from class.

It is known that over the last n days, Morty received n grades (one per day), which form the array $a_1, a_2, \dots, a_n$.

Rick wants to prove to Morty that he did better. In his time, over the same n days, Rick also received n grades, which form the array $b_1, b_2, \dots, b_n$ of numbers from 0 to 5.

Rick is very cunning, so he wants to rearrange his grades so that on each day his grade is **not less than** Morty's grade. Moreover, if after the rearrangement either Rick or Morty has a 0 in the gradebook for any day, then Rick **doesn't care** who received what grade on that day.

More formally, Rick wants to find an array c , which is obtained by rearranging the elements of array b , such that for each index i at least one of the following three conditions holds: $c_i \geq a_i$, $a_i = 0$, or $c_i = 0$.

Help Rick find a suitable order of elements from b or tell him that it is impossible.

Input

Each test consists of several sets of input data. The first line contains one integer t ($1 \leq t \leq 10^5$) — the number of input data sets. The description of the input data sets follows.

The first line of each input data set contains an integer n ($1 \leq n \leq 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \leq a_i \leq 5$) — Morty's grades.

The third line contains n integers $b_1, b_2, \dots, b_n$ ($0 \leq b_i \leq 5$) — Rick's grades.

It is guaranteed that the sum of n across all input data sets does not exceed $3 \cdot 10^5$.

Output

For each input data set, output the elements of array b in **any suitable** order on a single line.

If Rick cannot find the required rearrangement of grades, output -1 .

Examples

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

Problem J. Exercise for Dania

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

One Monday evening, Dania was sitting in the library solving programming problems. He loved spending time solving problems, but he loved even more to boast about the problems he had already solved in front of other guys. His acquaintance knew that Dania was a master at his craft and asked him to help with a simple exercise.

Dania was stumped by such *exercises*, but he didn't want to lose his reputation as an experienced programmer. So he turned to you for help. The task was as follows:

Consider the *function of the minimum non-trivial divisor* $M(x) = d$, where d is the smallest number greater than one that divides x . The number one has no such divisors, so it is defined that $M(1) = 0$.

For example, $M(12) = 2$, $M(35) = 5$, $M(179) = 179$.

For a given array $a_1, \dots, a_n$ of length n , it is required to compute the value of the following expression for each k from 1 to n :

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^k M\left(\frac{a_i a_j}{\gcd(a_i, a_j)^2}\right)$$

Here, $\gcd(x, y)$ denotes the *greatest common divisor* (*GCD*) of the numbers x and y .

Input

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

The first line of each input data set contains a single integer n ($1 \leq n \leq 150\,000$) — the size of the array.

The next line of each input data set contains n numbers a_i ($2 \leq a_i \leq 1\,000\,000$) — the array a itself.

It is guaranteed that the sum of n across all sets does not exceed 150 000.

Output

For each input data set, output n numbers — the value of the desired sum for each $1 \leq k \leq n$.

Example

standard input	standard output
4	0 2 6
3	0 2 6 13 22
2 4 8	0 2 6 13 19 30 54 87 113 133
5	0 999749 2999457 5999040 9997724
2 3 4 5 6	
10	
10 12 15 20 10 13 17 11 25 30	
5	
999979 999749 999959 999917 999671	

Problem K. Squirrel and Steps

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 512 megabytes

The evil owner Sasha kicked his cat Squirrel out onto the street. Now Squirrel needs to get back home. Sasha lives on the second floor of his palace. To reach him, Squirrel must go through n steps. Initially, Squirrel is on step number 0, and she needs to get to step number n . Squirrel is not very long, so the distance between any two of her paws must not exceed 2. In one step, the cat moves one of her paws to the next step.



More formally, this means that at any moment for any positions of her paws l_i and l_j , $|l_i - l_j| \leq 2$.

Like any cat, Squirrel has different paws: left front, right front, left back, right back, and they are distinguishable. You need to help Squirrel count the number of ways to climb to step n . Two ways are considered different if at some step Squirrel moved different paws. Initially, all paws are on step number 0.

Since the number of ways can be quite large, output it modulo $10^9 + 7$.

Input

The first line contains one integer t ($1 \leq t \leq 1000$) — the number of test cases.

In a single line of each test case, there is one integer n ($1 \leq n \leq 1\,000\,000$) — the number of steps Squirrel needs to go through.

Note that there is no additional restriction on the sum of n across all test cases.

Output

For each test case, output one number — the number of ways to reach step n modulo $10^9 + 7$.

Example

standard input	standard output
3	24
1	2520
2	151698586
6	

Problem L. Vector Magic

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

Everyone knows about the MOP — the International Society of Primates. One of the members of this society, the monkey Kirill, is studying magic.



Today, while working as a janitor in the stable, he found a strange piece of paper with n integers $a_1, \dots, a_n$ written on it. He immediately realized that this was a magical spell! To use it, he needs to place a magical rune next to each number and snap his fingers. After that, the paper will disappear and $\frac{(a,b)}{\|b\|_2}$ bananas will appear, where a — a vector in n -dimensional space consisting of the numbers from the paper, and b — a vector in the same space consisting of the powers of the magical runes placed next to the corresponding numbers on the paper.

Kirill has been studying magic for many years, so he already knows the values of these formulas, but he has not studied mathematics, and therefore he asks you to help him. You need to compute the value of the following expression:

$$\frac{(a,b)}{\|b\|_2} = \frac{a_1 \cdot b_1 + a_2 \cdot b_2 + \dots + a_n \cdot b_n}{\sqrt{b_1^2 + b_2^2 + \dots + b_n^2}}$$

Kirill also does not know what all this means, it's just magic

Unfortunately, Kirill only has runes with powers $-3, -1, 1, 3$, and he has n of each, meaning he can create any vector b consisting of these numbers. Help Kirill distribute the runes optimally to get as many bananas as possible!

Input

The first line contains one integer n ($1 \leq n \leq 10^4$) — the number of numbers on the paper.

The second line contains n integers a_i ($-300 \leq a_i \leq 300$) — the numbers written on the paper.

Output

Output the optimal vector (the best choice of runes) $b_1, \dots, b_n$, where $b_i \in \{-3, -1, 1, 3\}$.

If there are multiple optimal vectors, you can output any of them.

Examples

standard input	standard output
3 1 1 -1	1 1 -1
3 -3 20 1	-1 3 1
5 0 0 0 0 0	-1 -1 -1 -1 -1