A. New Year and Naming

time limit per test: 1 second
memory limit per test: 1024 megabytes
input: standard input
output: standard output

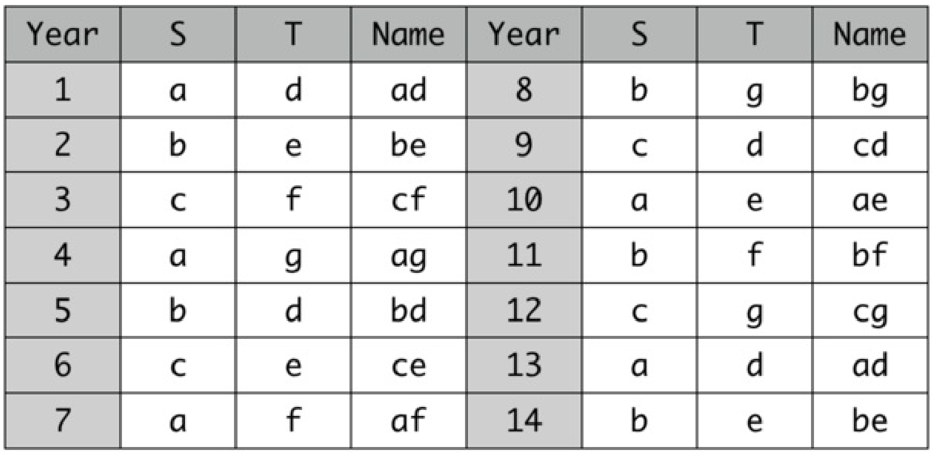
Happy new year! The year 2020 is also known as *Year Gyeongja* (, *gyeongja-nyeon*) in Korea. Where did the name come from? Let's briefly look at the *Gapja* system, which is traditionally used in Korea to name the years.

There are two sequences of n strings $s_1, s_2, s_3, \ldots, s_n$ and m strings $t_1, t_2, t_3, \ldots, t_m$. These strings contain only lowercase letters. There might be duplicates among these strings.

Let's call a concatenation of strings x and y as the string that is obtained by writing down strings x and y one right after another without changing the order. For example, the concatenation of the strings "`code`" and "`forces`" is the string "`codeforces`".

The year 1 has a name which is the concatenation of the two strings s_1 and t_1. When the year increases by one, we concatenate the next two strings in order from each of the respective sequences. If the string that is currently being used is at the end of its sequence, we go back to the first string in that sequence.

For example, if $n = 3, m = 4, s =${"`a`", "`b`", "`c`"}, $t =${"`d`", "`e`", "`f`", "`g`"}, the following table denotes the resulting year names. Note that the names of the years may repeat.



Year	S	T	Name	Year	S	T	Name
1	a	d	ad	8	b	g	bg
2	b	e	be	9	c	d	cd
3	c	f	cf	10	a	e	ae
4	a	g	ag	11	b	f	bf
5	b	d	bd	12	c	g	cg
6	c	e	ce	13	a	d	ad
7	a	f	af	14	b	e	be

You are given two sequences of strings of size n and m and also q queries. For each query, you will be given the current year. Could you find the name corresponding to the given year, according to the *Gapja* system?

Input

The first line contains two integers n, m ($1 \le n, m \le 20$).

The next line contains n strings $s_1, s_2, \ldots, s_n$. Each string contains only lowercase letters, and they are separated by spaces. The length of each string is at least 1 and at most 10.

The next line contains m strings $t_1, t_2, \ldots, t_m$. Each string contains only lowercase letters, and they are separated by spaces. The length of each string is at least 1 and at most 10.

Among the given $n + m$ strings may be duplicates (that is, they are not necessarily all different).

The next line contains a single integer q ($1 \le q \le 2\,020$).

In the next q lines, an integer y ($1 \le y \le 10^9$) is given, denoting the year we want to know the name for.

Output

Print q lines. For each line, print the name of the year as per the rule described above.

Example

input

```
10 12
sin im gye gap eul byeong jeong mu gi gyeong
yu sul hae ja chuk in myo jin sa o mi sin
14
1
2
3
4
10
11
```

```
12
13
73
2016
2017
2018
2019
2020
```

output

```
sinyu
imsul
gyehae
gapja
gyeongo
sinmi
imsin
gyeyu
gyeyu
byeongsin
jeongyu
musul
gihae
gyeongja
```

Note

The first example denotes the actual names used in the *Gapja* system. These strings usually are either a number or the name of some animal.

B. New Year and Ascent Sequence

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

A sequence $a = [a_1, a_2, \ldots, a_l]$ of length l has an **ascent** if there exists a pair of indices (i, j) such that $1 \le i < j \le l$ and $a_i < a_j$. For example, the sequence $[0, 2, 0, 2, 0]$ has an ascent because of the pair $(1, 4)$, but the sequence $[4, 3, 3, 3, 1]$ doesn't have an ascent.

Let's call a concatenation of sequences p and q the sequence that is obtained by writing down sequences p and q one right after another without changing the order. For example, the concatenation of the $[0, 2, 0, 2, 0]$ and $[4, 3, 3, 3, 1]$ is the sequence $[0, 2, 0, 2, 0, 4, 3, 3, 3, 1]$. The concatenation of sequences p and q is denoted as $p + q$.

Gyeonggeun thinks that sequences with ascents bring luck. Therefore, he wants to make many such sequences for the new year. Gyeonggeun has n sequences $s_1, s_2, \ldots, s_n$ which may have different lengths.

Gyeonggeun will consider all n^2 pairs of sequences s_x and s_y ($1 \le x, y \le n$), and will check if its concatenation $s_x + s_y$ has an ascent. Note that he may select the same sequence twice, and the order of selection matters.

Please count the number of pairs (x, y) of sequences $s_1, s_2, \ldots, s_n$ whose concatenation $s_x + s_y$ contains an ascent.

Input

The first line contains the number n ($1 \le n \le 100\,000$) denoting the number of sequences.

The next n lines contain the number l_i ($1 \le l_i$) denoting the length of s_i, followed by l_i integers $s_{i,1}, s_{i,2}, \ldots, s_{i,l_i}$ ($0 \le s_{i,j} \le 10^6$) denoting the sequence s_i.

It is guaranteed that the sum of all l_i does not exceed $100\,000$.

Output

Print a single integer, the number of pairs of sequences whose concatenation has an ascent.

Examples

input

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

output

```
9
```

input

```
3
4 2 0 2 0
6 9 9 8 8 7 7
1 6
```

output

```
7
```

input

```
10
3 62 24 39
1 17
1 99
1 60
1 64
1 30
2 79 29
2 20 73
2 85 37
1 100
```

output

```
72
```

Note

For the first example, the following 9 arrays have an ascent: $[1, 2], [1, 2], [1, 3], [1, 3], [1, 4], [1, 4], [2, 3], [2, 4], [3, 4]$. Arrays with the same contents are counted as their occurences.

C. New Year and Permutation

time limit per test: 1 second
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Recall that the permutation is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

A sequence a is a subsegment of a sequence b if a can be obtained from b by deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end. We will denote the subsegments as $[l, r]$, where l, r are two integers with $1 \le l \le r \le n$. This indicates the subsegment where $l - 1$ elements from the beginning and $n - r$ elements from the end are deleted from the sequence.

For a permutation $p_1, p_2, \ldots, p_n$, we define a *framed segment* as a subsegment $[l, r]$ where $\max\{p_l, p_{l+1}, \ldots, p_r\} - \min\{p_l, p_{l+1}, \ldots, p_r\} = r - l$. For example, for the permutation $(6, 7, 1, 8, 5, 3, 2, 4)$ some of its framed segments are: $[1, 2], [5, 8], [6, 7], [3, 3], [8, 8]$. In particular, a subsegment $[i, i]$ is always a framed segments for any i between 1 and n, inclusive.

We define the *happiness* of a permutation p as the number of pairs (l, r) such that $1 \le l \le r \le n$, and $[l, r]$ is a framed segment. For example, the permutation $[3, 1, 2]$ has happiness 5: all segments except $[1, 2]$ are framed segments.

Given integers n and m, Jongwon wants to compute the sum of happiness for all permutations of length n, modulo the prime number m. Note that there exist $n!$ (factorial of n) different permutations of length n.

Input

The only line contains two integers n and m ($1 \le n \le 250\,000$, $10^8 \le m \le 10^9$, m is prime).

Output

Print r ($0 \le r < m$), the sum of happiness for all permutations of length n, modulo a prime number m.

Examples

input

```
1 993244853
```

output

```
1
```

input

```
2 993244853
```

output

```
6
```

input

```
3 993244853
```

output

```
32
```

input

```
2019 993244853
```

output

```
923958830
```

input

```
2020 437122297
```

output

```
265955509
```

Note

For sample input $n = 3$, let's consider all permutations of length 3:

- $[1, 2, 3]$, all subsegments are framed segment. Happiness is 6.
- $[1, 3, 2]$, all subsegments except $[1, 2]$ are framed segment. Happiness is 5.
- $[2, 1, 3]$, all subsegments except $[2, 3]$ are framed segment. Happiness is 5.
- $[2, 3, 1]$, all subsegments except $[2, 3]$ are framed segment. Happiness is 5.
- $[3, 1, 2]$, all subsegments except $[1, 2]$ are framed segment. Happiness is 5.
- $[3, 2, 1]$, all subsegments are framed segment. Happiness is 6.

Thus, the sum of happiness is $6 + 5 + 5 + 5 + 5 + 6 = 32$.

D. New Year and Conference

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Filled with optimism, Hyunuk will host a conference about how great this new year will be!

The conference will have n lectures. Hyunuk has two candidate venues a and b. For each of the n lectures, the speaker specified two time intervals $[sa_i, ea_i]$ ($sa_i \le ea_i$) and $[sb_i, eb_i]$ ($sb_i \le eb_i$). If the conference is situated in venue a, the lecture will be held from sa_i to ea_i, and if the conference is situated in venue b, the lecture will be held from sb_i to eb_i. Hyunuk will choose one of these venues and **all** lectures will be held at that venue.

Two lectures are said to overlap if they share any point in time in common. Formally, a lecture held in interval $[x, y]$ overlaps with a lecture held in interval $[u, v]$ if and only if $\max(x, u) \le \min(y, v)$.

We say that a participant can *attend* a subset s of the lectures if the lectures in s do not pairwise overlap (i.e. no two lectures overlap). Note that the possibility of attending may depend on whether Hyunuk selected venue a or venue b to hold the conference.

A subset of lectures s is said to be *venue-sensitive* if, for one of the venues, the participant can attend s, but for the other venue, the participant cannot attend s.

A venue-sensitive set is problematic for a participant who is interested in attending the lectures in s because the participant cannot be sure whether the lecture times will overlap. Hyunuk will be happy if and only if there are no venue-sensitive sets. Determine whether Hyunuk will be happy.

Input

The first line contains an integer n ($1 \le n \le 100\,000$), the number of lectures held in the conference.

Each of the next n lines contains four integers sa_i, ea_i, sb_i, eb_i ($1 \le sa_i, ea_i, sb_i, eb_i \le 10^9$, $sa_i \le ea_i, sb_i \le eb_i$).

Output

Print "YES" if Hyunuk will be happy. Print "NO" otherwise.

You can print each letter in any case (upper or lower).

Examples

input

```
2
1 2 3 6
3 4 7 8
```

output

```
YES
```

input

```
3
1 3 2 4
4 5 6 7
3 4 5 5
```

output

```
NO
```

input

```
6
1 5 2 9
2 4 5 8
3 6 7 11
7 10 12 16
8 11 13 17
9 12 14 18
```

output

```
YES
```

Note

In second example, lecture set $\{1, 3\}$ is venue-sensitive. Because participant can't attend this lectures in venue a, but can attend in venue b.

In first and third example, venue-sensitive set does not exist.

E. New Year and Castle Construction

time limit per test: 3 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Kiwon's favorite video game is now holding a new year event to motivate the users! The game is about building and defending a castle, which led Kiwon to think about the following puzzle.

In a 2-dimension plane, you have a set $s = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ consisting of n distinct points. In the set s, **no three distinct points lie on a single line**. For a point $p \in s$, we can protect this point by building a castle. A **castle** is a simple quadrilateral (polygon with 4 vertices) that strictly encloses the point p (i.e. the point p is strictly inside a quadrilateral).

Kiwon is interested in the number of 4-point subsets of s that can be used to build a castle protecting p. Note that, if a single subset can be connected in more than one way to enclose a point, it is counted only once.

Let $f(p)$ be the number of 4-point subsets that can enclose the point p. Please compute the sum of $f(p)$ for all points $p \in s$.

Input

The first line contains a single integer n ($5 \le n \le 2\,500$).

In the next n lines, two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) denoting the position of points are given.

It is guaranteed that all points are distinct, and there are no three collinear points.

Output

Print the sum of $f(p)$ for all points $p \in s$.

Examples

input

```
5
-1 0
1 0
-10 -1
10 -1
0 3
```

output

```
2
```

input

```
8
0 1
1 2
2 2
1 3
0 -1
-1 -2
-2 -2
-1 -3
```

output

```
40
```

input

```
10
```

```
588634631 265299215
-257682751 342279997
527377039 82412729
145077145 702473706
276067232 912883502
822614418 -514698233
280281434 -41461635
65985059 -827653144
188538640 592896147
-857422304 -529223472
```

output

```
213
```

F. New Year and Social Network

time limit per test: 4 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Donghyun's new social network service (SNS) contains n users numbered $1, 2, \ldots, n$. Internally, their network is a *tree graph*, so there are $n-1$ direct connections between each user. Each user can reach every other users by using some sequence of direct connections. From now on, we will denote this primary network as T_1.

To prevent a possible server breakdown, Donghyun created a backup network T_2, which also connects the same n users via a tree graph. If a system breaks down, exactly one edge $e \in T_1$ becomes unusable. In this case, Donghyun will protect the edge e by picking another edge $f \in T_2$, and add it to the existing network. This new edge should make the network be connected again.

Donghyun wants to assign a replacement edge $f \in T_2$ for as many edges $e \in T_1$ as possible. However, since the backup network T_2 is fragile, $f \in T_2$ can be assigned as the replacement edge for at most one edge in T_1. With this restriction, Donghyun wants to protect as many edges in T_1 as possible.

Formally, let $E(T)$ be an edge set of the tree T. We consider a bipartite graph with two parts $E(T_1)$ and $E(T_2)$. For $e \in E(T_1), f \in E(T_2)$, there is an edge connecting $\{e, f\}$ if and only if graph $T_1 - \{e\} + \{f\}$ is a tree. You should find a maximum matching in this bipartite graph.

Input

The first line contains an integer n ($2 \le n \le 250\,000$), the number of users.

In the next $n-1$ lines, two integers a_i, b_i ($1 \le a_i, b_i \le n$) are given. Those two numbers denote the indices of the vertices connected by the corresponding edge in T_1.

In the next $n-1$ lines, two integers c_i, d_i ($1 \le c_i, d_i \le n$) are given. Those two numbers denote the indices of the vertices connected by the corresponding edge in T_2.

It is guaranteed that both edge sets form a tree of size n.

Output

In the first line, print the number m ($0 \le m < n$), the maximum number of edges that can be protected.

In the next m lines, print four integers a_i, b_i, c_i, d_i. Those four numbers denote that the edge (a_i, b_i) in T_1 is will be replaced with an edge (c_i, d_i) in T_2.

All printed edges should belong to their respective network, and they should link to distinct edges in their respective network. If one removes an edge (a_i, b_i) from T_1 and adds edge (c_i, d_i) from T_2, the network should remain connected. The order of printing the edges or the order of vertices in each edge does not matter.

If there are several solutions, you can print any.

Examples

input

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

output

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

input

```
5
1 2
2 4
```

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

output

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

input

```
9
7 9
2 8
2 1
7 5
4 7
2 4
9 6
3 9
1 8
4 8
2 9
9 5
7 6
1 3
4 6
5 3
```

output

```
8
4 2 9 2
9 7 6 7
5 7 5 9
6 9 4 6
8 2 8 4
3 9 3 5
2 1 1 8
7 4 1 3
```

G. Seollal

time limit per test: 3 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

It is only a few days until Seollal (Korean Lunar New Year), and Jaehyun has invited his family to his garden. There are kids among the guests. To make the gathering more fun for the kids, Jaehyun is going to run a game of hide-and-seek.

The garden can be represented by a $n \times m$ grid of unit cells. Some (possibly zero) cells are blocked by rocks, and the remaining cells are free. Two cells are neighbors if they share an edge. Each cell has up to 4 neighbors: two in the horizontal direction and two in the vertical direction.

Since the garden is represented as a grid, we can classify the cells in the garden as either "`black`" or "`white`". The top-left cell is black, and two cells which are neighbors must be different colors. Cell indices are 1-based, so the top-left corner of the garden is cell $(1, 1)$.

Jaehyun wants to turn his garden into a *maze* by placing some walls between two cells. Walls can only be placed between neighboring cells. If the wall is placed between two neighboring cells a and b, then the two cells a and b are not neighboring from that point. One can walk directly between two neighboring cells if and only if there is no wall directly between them.

A *maze* must have the following property. For each pair of free cells in the maze, there must be exactly one simple path between them. A simple path between cells a and b is a sequence of free cells in which the first cell is a, the last cell is b, all cells are distinct, and any two consecutive cells are neighbors which are not directly blocked by a wall.

At first, kids will gather in cell $(1, 1)$, and start the hide-and-seek game. A kid can hide in a cell if and only if that cell is free, it is not $(1, 1)$, and has exactly one free neighbor. Jaehyun planted roses in the black cells, so it's dangerous if the kids hide there. So Jaehyun wants to create a maze where the kids can only hide in white cells.

You are given the map of the garden as input. Your task is to help Jaehyun create a maze.

Input

Your program will be judged in multiple test cases.

The first line contains the number of test cases t. ($1 \le t \le 100$). Afterward, t test cases with the described format will be given.

The first line of a test contains two integers n, m ($2 \le n, m \le 20$), the size of the grid.

In the next n line of a test contains a string of length m, consisting of the following characters (without any whitespace):

- 0: A free cell.
- X: A rock.

It is guaranteed that the first cell (cell $(1, 1)$) is free, and every free cell is reachable from $(1, 1)$.

If $t \geq 2$ is satisfied, then the size of the grid will satisfy $n \leq 10, m \leq 10$. In other words, if any grid with size $n > 10$ or $m > 10$ is given as an input, then it will be the only input on the test case ($t = 1$).

Output

For each test case, print the following:

If there are no possible mazes, print a single line `NO`.

Otherwise, print a single line YES, followed by a grid of size $(2n-1) \times (2m-1)$ denoting the found maze. The rules for displaying the maze follows. All cells are indexed in 1-base.

- For all $1 \leq i \leq n, 1 \leq j \leq m$, if the cell (i, j) is free cell, print `'0'` in the cell $(2i-1, 2j-1)$. Otherwise, print `'X'` in the cell $(2i-1, 2j-1)$.
- For all $1 \leq i \leq n, 1 \leq j \leq m-1$, if the neighboring cell $(i, j), (i, j+1)$ have wall blocking it, print `' '` in the cell $(2i-1, 2j)$. Otherwise, print **any printable character except spaces** in the cell $(2i-1, 2j)$. A printable character has an ASCII code in range $[32, 126]$: This includes spaces and alphanumeric characters.
- For all $1 \leq i \leq n-1, 1 \leq j \leq m$, if the neighboring cell $(i, j), (i+1, j)$ have wall blocking it, print `' '` in the cell $(2i, 2j-1)$. Otherwise, print **any printable character except spaces** in the cell $(2i, 2j-1)$
- For all $1 \leq i \leq n-1, 1 \leq j \leq m-1$, print **any printable character** in the cell $(2i, 2j)$.

Please, be careful about trailing newline characters or spaces. Each row of the grid should contain **exactly** $2m-1$ characters, and rows should be separated by a newline character. Trailing spaces must not be omitted in a row.

Example

input

```
4
2 2
OO
OO
3 3
OOO
XOO
OOO
4 4
OOOX
XOOX
OOXO
OOOO
5 6
OOOOOO
OOOOOO
OOOOOO
OOOOOO
OOOOOO
```

output

```
YES
OOO
 O
OOO
NO
YES
OOOOO X
 O O
X O O X
 O
OOO X O
O O   O
O OOOOO
YES
OOOOOOOOOOO
 O   O   O
OOO OOO OOO
O   O   O
OOO OOO OOO
 O   O   O
OOO OOO OOO
O   O   O
OOO OOO OOO
```