Problem A: Rubik's Rectangle

A new puzzle which aims to conquer the game market is a fusion of Rubik's Cube and Fifteen. The board is an $H \times W$ frame with tiles with all numbers from 1 to $H \cdot W$ printed on them.

1	2	15	4
8	7	11	5
12	6	10	9
13	14	3	16

The only type of move that is allowed is *flipping* either one of the rows or one of the columns. Flipping reverses the order of the row's (or column's) elements. Below the third row is flipped:

1	2	15	4
8	7	11	5
9	10	6	12
13	14	3	16

You are given a board with tiles numbered in some arbitrary order. Determine a sequence of flips that brings the board to the nicely sorted position, if possible.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

The description of each test case starts with an empty line. The next line contains two space-separated integers W and H ($1 \leqslant W, H \leqslant 100$) – the width and height of the puzzle, respectively. Each of the next H lines contains W space-separated integers – the numbers printed on consecutive tiles.

Output

Print the answers to the test cases in the order in which they appear in the input. Start the output for each test case with the word **POSSIBLE** or **IMPOSSIBLE**, depending on whether it is possible to solve the puzzle. If a solution exists, print (in the same line) first the number of moves (possibly 0) and then their descriptions, each consisting of a single letter **R** or **C** specifying whether we are to flip a row or a column, concatenated with the index of the row or column to flip.

Any solution will be accepted as long as it does not use more than $10 \cdot W \cdot H$ moves. Each test case is either solvable within this limit, or not solvable at all.

Example

For an example input	a possible correct answer is:

<pre>
4
 	
3 3
1 2 3
4 5 6
9 8 7
 	
4 2
1 2 3 4
5 6 7 8
 	
4 4
1 2 15 4
8 7 11 5
12 6 10 9
13 14 3 16
 	
3 4
1 2 4
3 5 6
7 8 9
10 11 12</pre>
	<pre>
	POSSIBLE 1 R3
	POSSIBLE 0
	POSSIBLE 3 R3 C3 R2
	IMPOSSIBLE</pre>

Problem B: What does the fox say?

Determined to discover the ancient mystery – the sound that the fox makes – you went into the forest, armed with a very good digital audio recorder. The forest is, however, full of animals' voices, and on your recording, many different sounds can be heard. But you are well prepared for your task: you know exactly all the sounds which other animals make. Therefore the rest of the recording – all the unidentified noises – must have been made by the fox.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

The first line of each test case contains the recording – words over lower case English alphabet, separated by spaces. Each contains at most 100 letters and there are no more than 100 words. The next few lines are your pre-gathered information about other animals, in the format <**animal**> **goes** <**sound**>. There are no more than 100 animals, their names are not longer than 100 letters each and are actual names of animals in English. There is no **fox goes ...** among these lines.

The last line of the test case is exactly the question you are supposed to answer: *what does the fox say?*

Output

For each test case, output one line containing the sounds made by the fox, in the order from the recording. You may assume that the fox was not silent (contrary to popular belief, foxes do not communicate by Morse code).

Example

For an example input

```
1
toot woof wa ow ow ow pa blub blub pa toot pa blub pa pa ow pow toot
dog goes woof
fish goes blub
elephant goes toot
seal goes ow
what does the fox say?
```

the correct answer is:

```
wa pa pa pa pa pa pow
```

Problem C: Magical GCD

The *Magical GCD* of a nonempty sequence of positive integers is defined as the product of its length and the greatest common divisor of all its elements.

Given a sequence $(a_1, ..., a_n)$, find the largest possible Magical GCD of its connected subsequence.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

The description of each test case starts with a line containing a single integer n, $1 \leqslant n \leqslant 100\,000$. The next line contains the sequence $a_1, a_2, ..., a_n$, $1 \leqslant a_i \leqslant 10^{12}$.

Output

For each test case output one line containing a single integer: the largest Magical GCD of a connected subsequence of the input sequence.

Example

For an example input	the correct answer is:
1 5 30 60 20 20 20	80

Problem D: Subway

Johny is going to visit his friend Michelle. His dad allowed him to go there on his own by subway. Johny loves traveling by subway and would gladly use this opportunity to spend half a day underground, but his dad obliged him to make as few line changes as possible. There are a lot of stations in the city, and several subway lines connecting them. All trains are perfectly synchronized – the travel between two consecutive stations on every line takes exactly one minute, and changing lines at any station takes no time at all.

Given the subway map, help Johny to plan his trip so that he can travel for as long as possible, while still following his dad's order.

Input

First line of input contains the number of test cases T. The descriptions of the test cases follow:

The description of each test case starts with an empty line. The next two lines begin with the strings **Stops:** and **Lines:**, and contain the names (separated by a comma and a space) of all subway stops and lines, respectively. A single line for each subway line follows (in no particular order), beginning with <**line-name**> **route:** and listing the names of the stops along this particular line. The final two lines specify the names of the (different) stations nearby Johny's and Michelle's homes.

In each test case, there are at most 300 000 stations and 100 000 lines, whose total length does not exceed 1 000 000. The names of lines and stations are between 1 and 50 characters long and can contain letters, digits, hyphens (-), apostrophes (') and "and" signs (&). All lines are bidirectional (although changing the direction of travel counts as a line change) and there are no self-crossings.

Output

Print the answers to the test cases in the order in which they appear in the input. For each test case, print a single line summarizing the optimal route Johny can take (see example output for exact format). You may assume that such a route always exists.

Example

Some lines in the example test data below were too long and had to be wrapped. You can access full sample tests at your workstation.

For an example input

```
3

Stops: OxfordCircus, PiccadillyCircus, HydeParkCorner, King'sCross,
GreenPark, Arsenal, Victoria, Highbury&Islington, LeicesterSquare
Lines: Blue, Cyan
Cyan route: Highbury&Islington, King'sCross, OxfordCircus,
GreenPark, Victoria
Blue route: HydeParkCorner, GreenPark, PiccadillyCircus,
LeicesterSquare, King'sCross, Arsenal
Johny lives at King'sCross
Michelle lives at GreenPark

Stops: OxfordCircus, PiccadillyCircus, HydeParkCorner, King'sCross,
GreenPark, Arsenal, Victoria, Highbury&Islington, LeicesterSquare
Lines: Blue, Cyan
Cyan route: Highbury&Islington, King'sCross, OxfordCircus,
GreenPark, Victoria
Blue route: HydeParkCorner, GreenPark, PiccadillyCircus,
LeicesterSquare, King'sCross, Arsenal
Johny lives at PiccadillyCircus
Michelle lives at LeicesterSquare

Stops: OxfordCircus, PiccadillyCircus, HydeParkCorner, King'sCross,
GreenPark, Arsenal, Victoria, Highbury&Islington, LeicesterSquare
Lines: Blue, Cyan
Cyan route: Highbury&Islington, King'sCross, OxfordCircus,
GreenPark, Victoria
Blue route: HydeParkCorner, GreenPark, PiccadillyCircus,
LeicesterSquare, King'sCross, Arsenal
Johny lives at Victoria
Michelle lives at HydeParkCorner
```

the correct answer is:

```
optimal travel from King'sCross to GreenPark: 1 line, 3 minutes
optimal travel from PiccadillyCircus to LeicesterSquare: 1 line, 1 minute
optimal travel from Victoria to HydeParkCorner: 2 lines, 7 minutes
```

Problem E: Escape

You hit the emperor lich with full force and slay it. There is a stair leading upwards here. You climb upstairs. You drink from the pool. You feel much better. The karmic lizard punches through your armor and hits you. You die…

After an epic fight with the emperor lich, the hero struggles to escape the dungeon consisting of n chambers and $n-1$ corridors connecting them. He starts in chamber number 1 and must reach chamber number t, moving only along the corridors. All chambers are reachable from chamber number 1. Bruised after the last fight, the hero starts the journey with 0 hit-points (HP). These points represent his health – if ever they fall below zero, the hero's story ends there as a tragic one.

In some chambers there are monsters – a monster must be fought, and it always manages to take some of the hero's HP. In some other chambers there are magic pools – every pool restores some number of the hit-points. There is no upper limit on the hero's health. Every chamber can be visited multiple times, but the gain or loss of HP happens only once, on the very first visit.

Determine whether the hero can escape the dungeon alive.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

The first line of each test case contains two integers: the number of chambers n, $2 \leqslant n \leqslant 200\,000$, and the number of the exit chamber t, $2 \leqslant t \leqslant n$. The second line contains n space-separated integers between -10^6 and 10^6 – the i-th of them denotes the HP gain in the i-th chamber (negative denotes a monster, positive – a pool, and zero means that the chamber is empty). The first chamber does not contain a monster, but a pool is possible there. The exit chamber may contain a pool or a monster, and the monster will have to be fought before escaping.

The next $n-1$ lines contain the descriptions of corridors. Each one contains a pair of integers – the ends of a corridor.

Output

For each test case print a single line containing the word **escaped** if escape is possible, or **trapped** otherwise.

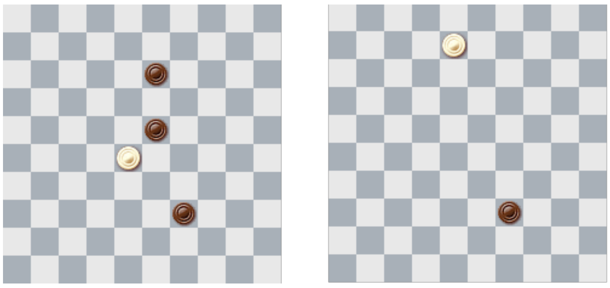
Example

For an example input	the correct answer is:
2 7 7 0 -3 2 2 3 -4 0 1 2 2 3 2 4 1 5 5 6 6 7 3 2 3 3 -4 1 3 2 3	escaped trapped

Problem F: Draughts

Draughts (or *checkers*) is a game played by two opponents, on opposite sides of a 10×10 board. The board squares are painted black and white, as on a classic chessboard. One player controls the dark, and the other the light pieces. The pieces can only occupy the black squares. The players make their moves alternately, each moving one of his own pieces.

The most interesting type of move is *capturing*: if a diagonally adjacent square contains an opponent's piece, it may be captured (and removed from the game) by jumping over it to the unoccupied square immediately beyond it. It is allowed to make several consecutive captures in one move, if they are all made with a single piece. It is also legal to capture by either forward or backward jumps.



The board before and after a single move with two captures.

You are given a draughts position. It is the light player's turn. Compute the maximal possible number of dark pieces he can capture in his next move.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case starts with an empty line. The following 10 lines of 10 characters each describe the board squares. The characters # and **.** denote empty black and white squares, **W** denotes a square with a light piece, **B** – a square with a dark piece.

Output

For each test case print a single line containing the maximal possible number of captures. If there is no legal move (for example, there are no light pieces on the board), simply output 0.

Example

For an example input	the correct answer is:
2 .#.#.#.#.# #.#.#.#.#. .#.#.B.#.# #.#.#.#.#. .#.#.B.#.# #.#.W.#.#. .#.#.#.#.# #.#.#.B.#. .#.#.#.#.# #.#.#.#.#. .#.#.#.#.# #.#.#.#.#. .#.#.B.#.# #.B.#.B.#. .#.#.B.#.# #.B.W.#.#. .#.B.B.#.# #.#.#.#.#. .#.B.B.#.# #.#.#.#.#.	2 4

Problem G: History course

You are to give a series of lectures on important historical events, one event per lecture, in some order. Each event lasted for some time interval $[a_i, b_i]$. We say that two events are *related* if their intervals have a common point. It would be convenient to schedule lectures on related events close to each other. Moreover, lectures on unrelated events should be given in the order in which the events have taken place (if an event A preceded an unrelated event B, then the lecture on A should precede the lecture on B).

Find the minimum integer $k \geqslant 0$ and an order of the lectures such that any two related events are scheduled at most k lectures apart from each other (lectures number i and j are considered to be $|i - j|$ lectures apart).

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

The first line of each test case contains the number n, $1 \leqslant n \leqslant 50\,000$. Each of the next n lines contains two integers a_i and b_i, $-10^9 \leqslant a_i \leqslant b_i \leqslant 10^9$ – the ends of the i-th interval. The intervals are pairwise different.

Output

Print the answers to the test cases in the order in which they appear in the input. The first line of the answer to each test case should contain the minimum value of k. The next n lines should list the intervals (in the same format as in the input) in an order such that any two related events are scheduled at most k lectures apart. Remember to put any unrelated intervals in the proper order!

Example

For an example input	a possible correct answer is:

<pre>1
3
1 6
2 3
4 5</pre>
	<pre>1
	2 3
	1 6
	4 5</pre>

Problem H: Chain & Co.

Chain & Co. specializes in producing infinitely strong chains. Because of their high quality products, they are quickly gaining market share. This leads to new challenges, some of which they could have never imagined before. Like, for example, automatic verification of link endurance with a computer program, which you are supposed to write.

The company produces *links* of equal size. Each link is an infinitely thin square frame in three dimensions (made of four infinitely thin segments).

During tests all links are axis-aligned [1] and placed so that no two frames touch. To make a proper strength test, two sets of links A and B are forged so that every link of A is inseparable from every link of B (being inseparable means that they cannot be moved apart without breaking one of them).

You stumble upon some links (axis-aligned, pairwise disjoint). Are they in proper testing position? In other words, can they be divided into two non-empty sets A and B with the desired property?

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

The description of each test case starts with an empty line. The next line contains an integer n, $1 \leqslant n \leqslant 10^6$ – the number of links in the chain. Each of the next n lines contains 6 space-separated integers $x_i, y_i, z_i, x'_i, y'_i, z'_i$, all between -10^9 and 10^9 – the coordinates of two opposite corners of the i-th link.

Output

For each test case, print a single line containing the word **YES** if the set is in proper testing position, or **NO** otherwise.

[1] Axis-aligned means that all segments are parallel to either X, Y, or Z axis.

Example

For an example input	the correct answer is:
3 2 0 0 0 0 10 10 -5 5 15 5 5 25 5 0 0 0 0 10 10 -5 5 6 5 5 16 -5 5 -6 5 5 4 -5 6 5 5 16 5 -5 -6 5 5 4 5 3 0 0 0 3 0 -3 1 -1 -1 1 2 -4 -1 -2 -2 2 1 -2	NO YES YES

Problem I: Crane

There are n crates waiting to be loaded onto a ship. The crates are numbered $1, 2, \ldots, n$, the numbers determining the order of loading. Unfortunately, someone messed up the transit and the crates are standing in a row in an arbitrary order. As there is only limited space in the dock area, you must sort the crates by swapping some of them.

You are given a crane that works in the following way: you select a connected interval of crates of even length. The crane then exchanges the first half of the interval with the second half. The order inside both halves remains unchanged. Determine the sequence of crane moves that reorders the crates properly.

The crane's software has a bug: the move counter is a 9-based (not 10-based, as you might think) integer with at most 6 digits. Therefore, the crane stops working (and has to be serviced) after $9^6 = 531441$ moves. Your solution must fit within this limit.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case starts with an integer n, $1 \leqslant n \leqslant 10\,000$, denoting the number of crates. In the next line a permutation of numbers $\{1, 2, \ldots, n\}$ follows.

Output

For each test case print a single line containing m – the number of swaps – followed by m lines describing the swaps in the order in which they should be performed. A single swap is described by two numbers – the indices of the first and the last element in the interval to be exchanged. Do not follow the crane's strange software design – use standard decimal numeral system.

Example

For an example input	a possible correct answer is:
2 6 5 4 6 3 2 1 5 1 2 3 4 5	5 1 2 4 5 5 6 4 5 1 6 0

Problem J: Captain Obvious and the Rabbit-Man

"It's you, Captain Obvious!" – cried the evil Rabbit-Man – "you came here to foil my evil plans!"

"Yes, it's me." – said Captain Obvious.

"But... how did you know that I would be here, on 625 Sunflower Street?! Did you crack my evil code?"

"I did. Three days ago, you robbed a bank on 5 Sunflower Street, the next day you blew up 25 Sunflower Street, and yesterday you left quite a mess under number 125. These are all powers of 5. And last year you pulled a similar stunt with powers of 13. You seem to have a knack for Fibonacci numbers, Rabbit-Man."

"That's not over! I will learn... **arithmetics!**" *– Rabbit-Man screamed as he was dragged into custody – "You will* **never** *know what to expect... Owww! Not my ears, you morons!"*

"Maybe, but right now you are being arrested." – Captain added proudly.

Unfortunately, Rabbit-Man has now indeed learned some more advanced arithmetics. To understand it, let us define the sequence F_n (being not completely unlike the Fibonacci sequence):

$F_1 = 1,$
$F_2 = 2,$
$F_n = F_{n-1} + F_{n-2}$ for $n \geqslant 3$.

Rabbit-Man has combined all his previous evil ideas into one master plan. On the i-th day, he does a malicious act on the spot number $p(i)$, defined as follows:

$$p(i) = a_1 \cdot F_1^i + a_2 \cdot F_2^i + \ldots + a_k \cdot F_k^i.$$

The number k and the integer coefficients $a_1, \ldots, a_k$ are fixed. Captain Obvious learned k, but does not know the coefficients. Given $p(1), p(2), \ldots, p(k)$, help him to determine $p(k+1)$. To avoid overwhelmingly large numbers, do all the calculations modulo a fixed prime number M. You may assume that $F_1, F_2, \ldots, F_n$ are pairwise distinct modulo M. You may also assume that there always exists a unique solution for the given input.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

The first line of each test case contains two integers k and M, $1 \leqslant k \leqslant 4000$, $3 \leqslant M \leqslant 10^9$. The second line contains k space-separated integers – the values of $p(1), p(2), \ldots, p(k)$ modulo M.

Output

Print the answers to the test cases in the order in which they appear in the input. For each test case print a single line containing one integer: the value of $p(k+1)$ modulo M.

Example

For an example input	the correct answer is:
2 4 619 5 25 125 6 3 101 5 11 29	30 83

Explanation: the first sequence is simply $5^i \mod 619$, therefore the next element is $5^5 \mod 619 = 30$. The second sequence is $2 \cdot 1^i + 3^i \mod 101$.

Problem K: Digraphs

A *digraph* is a graph with orientation... oh, sorry, not this time. Let's stop being nerds for a minute and talk about languages (*human* languages, not PHP).

Digraphs are pairs of characters that represent one phoneme (sound). For example, "ch" in English (as in "church") is a single consonant sound. The languages of Central Europe are fond of digraphs: Hungarian "sz", Czech "ch" and Polish "rz" are fine examples of them.

Digraphs are very annoying for people who do not use them natively. We will make up a letter-puzzle specifically for those people. Given a list of digraphs, construct a biggest possible square of lower case English letters such that its rows and columns *do not* contain any of these digraphs. This means that no two consecutive letters (read from top to bottom or from left to right) can form a digraph.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case starts with an integer n, $0 \leqslant n \leqslant 676$, denoting the number of forbidden digraphs. The n following lines contain the digraphs.

Output

For each test case print a square of the largest possible size which does not contain any of the digraphs. If it is possible to construct a square of size 20×20 or bigger, print only a 20×20 square.

Example

Part of the example test data below was omitted for clarity. You can access full sample tests at your workstation.

For an example input	a possible correct answer is:
2	aw
628	wz
aa	abababababababababab
az	babababababababababa
ba	abababababababababab
bb	babababababababababa
bc	abababababababababab
...	babababababababababa
by	abababababababababab
ca	babababababababababa
cb	abababababababababab
cc	babababababababababa
...	abababababababababab
cy	babababababababababa
da	abababababababababab
...	babababababababababa
dy	abababababababababab
...	babababababababababa
wa	abababababababababab
...	babababababababababa
wy	abababababababababab
ya	babababababababababa
...	
yy	
za	
zb	
...	
zy	
zz	
2	
aa	
bb	

Problem L: Bus

A bus with n passengers opens its door at the bus stop. Exactly half of its passengers and an additional half of a passenger get out. On the next stop, again, half of the passengers plus half of a passenger leave the bus. This goes on for k stops in total. Knowing that the bus leaves the last stop empty, and that no one was hurt during the trip, determine the initial number n of people in the bus.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

The only line of each test case contains the number of stops k, $1 \leqslant k \leqslant 30$.

Output

For each test case, output a single line containing a single integer – the initial number of bus passengers.

Example

For an example input	the correct answer is:
2 1 3	1 7