Problem A. Hello SCPC 2018!

Program:	`hello.(c|cpp|java)`
Input:	`hello.in`
Balloon Color:	`Orange`

Here are some facts about the problem set of this contest:

1. It consists of 12 problems written in English.
2. The first 4 problems are the easiest according to their estimated difficulty.
3. The first 4 problems are sorted in increasing order of their estimated difficulty.
4. You should probably pause now and tell your teammates, especially the one who's reading problem K.
5. The other 8 problems are harder than the first four, and they are ordered randomly.
6. Our estimations for the difficulty of the problems may not apply to all contestants. So make sure to read all the problems and find the ones that are the most suitable for you and your team.

You are given the difficulties of 12 problems of some contest in their exact order in the problem set, from A to L. Determine whether they are ordered in the same way we ordered this problem set or not.

Input

The first line of input contains a single integer T ($1 \leq T \leq 4096$), the number of test cases.

Each of the following T lines represents one test case, and contains 12 **distinct** space-separated integers d_1, d_2, ..., d_{12} ($1 \leq d_i \leq 100$), where d_i is the estimated difficulty of the i^{th} problem. A smaller d_i value represents an easier problem.

No two problems have the same estimated difficulty.

Output

For each test case, print a single line with "yes" if the given problems are ordered in the same way as in this contest, otherwise print "no".

Example

hello.in	Standard Output
3 1 2 4 8 32 16 35 99 78 50 64 85 1 2 3 4 5 6 7 8 9 10 11 12 4 3 2 1 5 6 7 8 9 10 11 12	yes yes no

Problem B. Binary Hamming

Program:	`hamming.(c|cpp|java)`
Input:	`hamming.in`
Balloon Color:	`Cyan`

The hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. For example, the hamming distance of strings "0101" and "0011" is 2 since they differ at two positions.

You are given two binary strings of length n. You can rearrange the characters in both strings. What is the maximum possible hamming distance between the two strings you can achieve?

Input

The first line of input contains a single integer T ($1 \leq T \leq 512$), the number of test cases.

The first line of each test case contains a single integer n ($1 \leq n \leq 100$), the length of the two strings.

Each of the following two lines contains a binary string of length n, where every character in both strings is either '0' or '1'.

Output

For each test case, output, on a single line, the maximum possible hamming distance between the two strings after rearranging them.

Example

hamming.in	Standard Output
3 4 0101 1001 3 000 101 4 0111 0111	4 2 2

Note

For the first test case, one possible way to achieve the maximum hamming distance is by rearranging the first string to "0110" and the second string to "1001", which results in the two strings differing in all 4 positions.

For the second test case, no matter how you rearrange the strings, the maximum hamming distance is 2.

For the third test case, a possible way to achieve the maximum distance 2 is by rearranging the strings to "0111" and "1101".

Problem C. Portals

Program:	`portals.(c|cpp|java)`
Input:	`portals.in`
Balloon Color:	`Purple`

Mr. Light is making a video game that can be represented as a ninja moving around a grid of 1 row and n columns.

Each cell is one of 5 types:

1. '.'; an empty cell. A ninja in this cell can move to the cell directly to its left or its right (if it exists and it is not blocked).
2. '#'; a blocked cell. A ninja cannot enter this cell.
3. 'o'; a portal. A ninja in this cell can jump to any other cell of the same type, or to the cell directly to its left or its right (if it exists and it is not blocked).
4. 's'; starting cell of the ninja. There is exactly one cell of this type in the grid.
5. 'e'; ending cell of the ninja. There is exactly one cell of this type in the grid.

The starting and ending cells are also empty cells, so the ninja can move from and to them in the same way.

Mr. Light wants to change a minimum number of empty cells to blocked cells (type 1 to type 2) such that there is no possible way for the ninja to get from the starting cell to the ending cell. Cells of type 2, 3, 4, and 5 cannot be changed.

Input

The first line of input contains a single integer T ($1 \leq T \leq 27000$), the number of test cases.

The first of each test case is n ($2 \leq n \leq 2 \times 10^5$), the number of columns in the grid.

The next line contains n characters, where each character is one of the following: '.', '#', 'o', 's', or 'e'. It is guaranteed that there is exactly one 's' and one 'e' in the grid.

The sum of n over all test cases doesn't exceed 2×10^6.

Output

For each test case, output the minimum number of empty cells that need to be blocked to make it impossible for the ninja to reach the ending cell. If there's no way to achieve that, print -1, on a single line.

Example

portals.in	Standard Output
3 9 o.s...e.o 11 #..soe..#.. 3 s#e	2 -1 0

Problem D. Carnival Slots

Program:	`balls.(c|cpp|java)`
Input:	`balls.in`
Balloon Color:	`Gold`

You go to the carnival and come across a nice little game. The carnival worker shows you the setup of the game, which can be represented as a 2-dimensional grid g with r rows and c columns. You are given the opportunity to change around the grid and maximize your score before the worker drops several balls into each column.

A cell in the i^{th} row (from top) and the j^{th} column (from left) is denoted by (i, j), and can be one of three different types of cells:

1. '.'; a ball that enters this cell will go to cell $(i + 1, j)$.
2. '\'; a ball that enters this cell will go to cell $(i + 1, j + 1)$.
3. '/'; a ball that enters this cell will go to cell $(i + 1, j - 1)$.

You may change a cell of type 2 and 3 to any of the three types, and you can change as many cells as you want. Cells of type 1 can't be changed.

Under the grid is aligned c buckets, where the i^{th} bucket is below the i^{th} column.

Each of the c buckets contains a score. For every ball that falls into a bucket, the score on that bucket is added to your total score and that ball stops. A ball that doesn't fall into a bucket gets a score of 0.

You are given how many balls the worker will drop into each column. Balls are dropped one after the another such that no two balls will collide. After making the changes, what is the maximum score you can achieve?

Your only condition for the grid g is that it's not allowed to have two adjacent cells in one row such that the left one is '\' and the right one is '/'.

Input

The first line of input contains a single integer T ($1 \le T \le 5300$), the number of test cases.

The first line of each test case contains two space-separated integers r and c ($1 \le r, c \le 500$), the dimensions of the grid.

The following line contains c space-separated integers b_1, b_2, ..., b_c ($0 \le b_i \le 10^8$), where b_i is the number of balls dropped into the i^{th} column.

Each of the following r lines contains c characters, representing the grid. Each character is either '.', '\', or '/'.

The last line of each test case contains c space-separated integers s_1, s_2, ..., s_c ($0 \le s_i \le 10^8$), where s_i is the score added after one ball drops into the i^{th} bucket.

The sum of $r \times c$ over all test cases doesn't exceed 4×10^6.

Output

For each test case, print on a single line the maximum score possible.

Example

balls.in	Standard Output
2 3 3 1 2 1 ../ ./. \.. 10 5 20 1 2 100000000 100000000 .. 1 100000000	70 10000000100000000

Problem E. 2Nodes

Program:	`nodes.(c|cpp|java)`
Input:	`nodes.in`
Balloon Color:	`White`

Mr. Light is playing an online game which involves a connected undirected graph. Each node is colored in either white, red, or blue.

When Mr. Light presses the start button, at every second, every white node with a non-white adjacent node will be colored in the same color as that adjacent node. If there are multiple such adjacent nodes, the color of the one with the minimum index is chosen. The game ends when there are no longer any white nodes. All color changes in the same second occur simultaneously.

Before pressing the start button, Mr. Light is given the opportunity to choose **a maximum of** two non-white nodes, and color them white. Can you help him choose **at most** two nodes such that when the game ends, the number of nodes that are colored red is maximized?

Input

The first line of input contains a single integer T ($1 \leq T \leq 1024$), the number of test cases.

The first line of each test case contains two integers n and m ($1 \leq n \leq 10^5$) ($n - 1 \leq m \leq 10^5$), the number of nodes and the number of edges, respectively.

The second line contains n space-separated integers c_i ($0 \leq c_i \leq 2$), where $c_i = 0$ if the i^{th} node is white, $c_i = 1$ if it is red, and $c_i = 2$ if it is blue. It is guaranteed that there's at least one red node.

Each of the following m lines contains two integers u and v ($1 \leq u, v \leq n, u \neq v$), representing an undirected edge between the nodes u and v.

It is guaranteed that the graph is connected and contains no self-loops or multiple edges.

The sum of m over all test cases doesn't exceed 5×10^6.

Output

For each test case, output a single line with the maximum number of nodes that will be colored red after changing the color of **at most** two non-white nodes to white.

Example

nodes.in	Standard Output
2 1 0 1 8 12 0 2 2 2 0 1 1 0 4 6 7 5 5 8 3 2 1 2 5 4 2 5 2 7 3 7 6 2 8 4 1 4	1 5

Problem F. Pretests

Program:	`tests.(c|cpp|java)`
Input:	`tests.in`
Balloon Color:	`Pink`

In a contest like the one you are participating in currently, a verdict of Accepted means that your solution was run on all test cases and produced the correct output for them all. However, some popular websites that hold online rounds prefer to use a pretest system. Your solution would be run on a subset of the test cases when you submit it, otherwise known as pretests.

For a problem with t pretests, solutions are judged on them in order, one by one, until it fails to produce a correct output. So, if a solution fails on the k^{th} test, it will request k test runs, and if it never fails, it will request t test runs.

To prevent the system from lagging, it would be preferable if the total number of test runs for all submissions was minimized.

You are given the verdicts of n submissions on each of the t pretest cases.

Print the minimum total number of runs needed if the pretests were ordered optimally. If there is more than one order that minimizes the answer, print the lexographically smallest one.

Input

The first line of input contains a single integer P, the number of problems you are ordering the pretests for.

The first line for each of the P problems contains two space-separated integers t and n $(1 \leq t \leq 20)$ $(1 \leq n \leq 2 \times 10^5)$, the number of pretests for this problem and the number of submissions, respectively.

Each of the following n lines contains t characters representing the verdicts of this submission on each of the pretests, the i^{th} character is '1' if this submission passes the i^{th} pretest, or '0' if it fails.

Output

For each of the P problems, output two lines. On the first line, print the minimum total number of test runs needed if the pretests were ordered optimally.

On the second line, print a permutation of the numbers from 1 to t, representing the lexographically smallest order of the pretests that produces the minimum answer.

Example

tests.in	Standard Output
1 4 3 1100 0101 0011	4 1 3 2 4

Note

A permutation a is lexicographically smaller than permutation b if there exists an index i such than $a_i < b_i$ and $a_j = b_j$ for all $(1 \leq j < i)$.

Problem G. Is Topo Logical?

Program:	`topo.(c|cpp|java)`
Input:	`topo.in`
Balloon Color:	`Silver`

A topologicial sort of a directed graph is a linear ordering of its vertices such that for every directed edge $u \to v$ going from u into v in the graph, u comes before v in the ordering. A commonly used example is a graph of courses, where the directed edge $u \to v$ represents that the course u must be taken before the course v. Hence making its topological sort a possible ordering of taking courses.

Mr. Topo has a directed graph. He calculated the in-degree for each node, which is equal to the number of edges going into the node, then he ran the following algorithm to obtain a topological ordering:

1. Find a node u with zero in-degree that was not processed before.
2. If no such u is found, terminate the algorithm.
3. Otherwise, subtract 1 from the in-degree of v for all nodes v where $u \to v$ is a directed edge in the graph. Repeat at step 1.

You will be given the initial in-degrees before running the algorithm, and the final in-degrees after the algorithm is terminated. Find a directed graph which would produce the given in-degree values before and after running the algorithm.

Your only conditions are that the graph must **not** contain self-loops nor repeated edges. A self-loop is an edge from the node to itself $u \to u$. Note that edge $u \to v$ is different from $v \to u$.

Input

The first of input contains a single integer T ($1 \le T \le 370$), the number of test cases.

The first line of each test case contains a single the integer n ($1 \le n \le 2 \times 10^5$), the number of nodes in the graph.

The second line contains n space-separated integers a_1, a_2, ..., a_n ($0 \le a_i < n$), where a_i is the initial in-degree of node i.

The third line contains n space-separated integers b_1, b_2, ..., b_n ($0 \le b_i \le a_i$), where b_i is the final in-degree of node i.

It is gauranteed that $\sum_{i=1}^{n} a_i \le 2 \times 10^5$, for each test case.

The sum of n over all test cases doesn't exceed 2×10^6.

The sum of all a_i values over all test cases doesn't exceed 2×10^6.

Output

For each test case, if there is no valid graph, output only -1 on a single line. Otherwise, output the number of edges m on the first line, and list the m directed edges $u \to v$ ($1 \le u, v \le n$) of the graph in the following m lines.

If there is more than one valid solution, output any of them.

Example

topo.in	Standard Output
3 5 0 1 1 1 1 0 1 1 1 1 3 0 1 0 0 1 0 5 0 2 1 1 3 0 1 1 1 2	4 3 2 2 3 2 4 2 5 -1 7 3 2 2 3 1 2 2 4 1 5 2 5 3 5

Problem H. Bugged System

Program:	`bugged.(c|cpp|java)`
Input:	`bugged.in`
Balloon Color:	`Yellow`

Mr. Light is visiting a city with a "smart" metro system. Or so it seems ...

There are exactly n stations in a line, where the i^{th} station is located at a distance x_i from the beginning of the line. You can check into some station, travel between the stations as many times as you want in both directions, and check out from another station. The metro card will track the sum of distances you traveled and charge you accordingly once you check out of your destination station.

However, there seems to be a bug in the system; if you happen to check in and out from the same station, you will be charged 0 credit. This creates the possibility of a scenario where one person traveling from stations a to b, and another person traveling from stations b to a, they can now meet up at some station and swap their cards. Therefore when they arrive, they both will pay 0 credit. Check the explanation of the first sample test for another scenario.

A person can go to any number of stations and wait as long as they like. Two people that meet at the same station can swap their cards.

You are given the starting and destination stations for m people traveling along the metro. Is it possible for all m people to check out from their destination stations and pay 0 credit? If so, print the minimum total distance they must travel to achieve this.

Input

The first line of input contains a single integer T ($1 \leq T \leq 3700$), the number of test cases.

The first line of each test case contains two space-separated integers n and m ($2 \leq n, m \leq 2 \times 10^5$), the number of stations and the number of people that will be using the metro stations.

The second line contains n **distinct** space-separated integers x_1, x_2, ..., x_n ($0 \leq x_i \leq 2 \times 10^6$), where x_i is the distance of the i^{th} station from the beginning of the line.

Each of the following m lines contains two integers s_i and d_i ($1 \leq s_i, d_i \leq n$, $s_i \neq d_i$), representing the starting and destination stations of i^{th} person.

The sum of n over all test cases doesn't exceed 2×10^6, the same is true for m.

Output

For each test case, output on a single line with the minimum total distance all m people need to travel to pay 0 credit.

If it is not possible, output -1 on a line.

Example

bugged.in	Standard Output
2 3 3 10 50 25 1 2 2 3 3 1 4 2 1 10 5 3 1 2 4 3	80 -1

Note

First sample explanation:

There are three metro cards, checked in at stations s_1, s_2, and s_3 and with person p_1, p_2, and p_3, respectively.

Person p_1 goes to station s_2, and swaps cards with person p_2. Now person p_1 can check out from station s_2 with 0 credit (distance traveled 40).

Person p_2 now has the card that was checked in from station s_1 and goes to station s_3 with it. He swaps that card with person p_3 and checks out from station s_3 with the new card he has (distance traveled 25).

Person p_3 now goes to station s_1 with the card that was also checked in from station s_1 and checks out there with 0 credit (distance traveled 15).

Sum of traveled distances is $40 + 25 + 15 = 80$.

Problem I. Rise of the Robots

Program:	`robots.(c|cpp|java)`
Input:	`robots.in`
Balloon Color:	`Red`

You made a robot to compete in an international competition. The organizers require that each robot must complete a set of movements to test its reflexes and balance before the competition starts.

A robot will make n moves on a circular table of radius R centered at $(0, 0)$, each move is in a straight line. For the i^{th} move, the robot will move from its current location (x, y) to $(x + dx_i, y + dy_i)$.

Everything is set for your test run, but you are faced with one final problem; you realized that your robot is out of balance. If any part of it gets off the table, it will fall and you will lose the competition. Your robot will be represented as a circle with radius r, with its location being the center of the circle.

Can you figure out a possible starting location for the robot to complete all the movements without falling?

Input

The first line of input contains an integer T, the number of test cases.

The first line of each test case contains three integers, n, R and r ($1 \leq n \leq 250$, $1 \leq r < R \leq 10^5$), the number of moves, the radius of the table, and the radius of the robot.

The i^{th} of the following n lines contains two integers dx_i and dy_i ($|dx_i| + |dy_i| > 0$), representing the i^{th} move.

It is guaranteed that there is at least one valid starting position to place the robot on.

Output

For each test case, output two numbers (S_x, S_y) on a single line, the coordinates of a valid starting position.

Your answer is considered correct if by extending the radius of the table R by 10^{-4}, the robot will be strictly inside the table throughout all moves.

Example

robots.in	Standard Output
1 4 5 2 3 0 0 3 3 -3 -3 -3	-3.000000000 0.000000000

Problem J. Clarifications

Program:	`clar.(c|cpp|java)`
Input:	`clar.in`
Balloon Color:	`Light Green`

Based on a true story.

If you've ever hosted a contest online before, you would know that there are many generic types of clarifications sent from users. Clarifications like "what is the answer for this test case" and "please help, my rating will be ruined" are far too common. Assume there are k types of clarifications.

Hasan is hosting his first contest and it will last for m minutes. Him and the contest administrator, KAN, will be answering clarifications. Hasan decided to answer clarifications of a type only after KAN answers at least one of that type, since he wasn't sure how to handle each type of the clarifications.

Answering a clarification takes one minute. Also, each of KAN and Hasan can answer at most one clarification in one minute. If KAN answered the first clarification of some type at minute x, Hasan will answer clarifications of that type only in minute $x + 1$ or later. If the first clarification of a type was sent in minute y, it can be answered by KAN in the same minute y. There are no other restrictions, clarifications can be answered in any order.

Throughout the contest, n clarifications were sent. You will be given the time, in minutes, each clarification was sent and its type.

Find the earliest time KAN and Hasan can finish answering all the clarifications if they collaborated in a way that minimizes this time. Note that this time can be after the end of the contest.

Input

The first line of input contains a single integer T ($1 \le T \le 256$), the number of test cases.

The first line of each test case contains three space-separated integers m, n, and k ($1 \le m \le 10^5$), ($1 \le k \le n \le 10^5$), the duration of the contest, the number of clarifications, and the number of types of clarifications respectively.

Each of the following n lines contains two space-separated integers t_i and p_i ($1 \le t_i \le m, 1 \le p_i \le k$), the time of the i^{th} clarification and its type, respectively. The clarifications are given in arbitrary order.

It is guaranteed that each of the k types will appear at least once in input.

The total sum of each of m and n over all test cases doesn't exceed 3×10^6.

Output

For each test case, output a single line with the earliest time KAN and Hasan can finish answering clarifications.

Example

clar.in	Standard Output
2 2 3 2 1 2 2 2 1 1 1 4 1 1 1 1 1 1 1 1 1	2 3

Problem K. Tourists' Tour

Program:	`tour.(c|cpp|java)`
Input:	`tour.in`
Balloon Color:	`Black`

The mayor is preparing for the arrival of the ICPC participants. On their tour, they will pass by n towers, each with a **distinct** height.

The architects designed these towers in a special way. The towers are aligned in a single line and numbered from 1 to n from left to right. Each of them is connected by a bridge to the closest tower to its left with a greater height, if one exists, and also by another bridge to the closest tower to its right with a greater height, if it exists.

The mayor doesn't want to make their tour boring. Therefore in preparation, he wants to color the n towers such that there is no bridge that connects two towers of the same color.

Help the mayor find a valid way to color the n towers with the minimum number of colors.

Input

The first line of input contains a single integer T ($1 \leq T \leq 128$), the number of test cases.

The first line of each test case contains a single integer n ($1 \leq n \leq 10^6$), the number of towers.

The second line contains n space-separated **distinct** integers ($1 \leq h_i \leq 10^9$), the i^{th} integer represents the height of the i^{th} tower.

The sum of n over all test cases doesn't exceed 5×10^6.

Output

For each test case, output two lines. The first line should contain the minimum number of colors k needed to color the towers.

The second line should contain n space-separated integers that represent a valid way to color the towers. The i^{th} integer is the color of the i^th tower, and it must be between 1 and k (inclusive).

If there are multiple valid ways, print any of them.

Example

tour.in	Standard Output
1 5 7 1 9 5 2	3 2 3 1 2 1

Note

In the first sample test, the pairs of towers connected by bridges are: (1, 2), (1, 3), (2, 3), (3, 4), and (4, 5).

Problem L. Sad Meals

Program:	`meals.(c|cpp|java)`
Input:	`meals.in`
Balloon Color:	`Rose`

Based on a sad story.

Mr. Light is living abroad and unfortunately, he doesn't know how to cook much meals. The day he learns how to cook a new meal, he will cook that meal for that day. Starting from the next day and assuming he now knows how to cook x meals, he will then start circulating between them, cooking one meal per day in the order he learned them in; from 1 to x then back to 1 again, until he learns a new meal. On day 1, Mr. Light learns how to cook the first meal.

For example, here is a valid sequence of meals he ate during the span of his first 15 days, assuming he learned to cook a new meal on days 1, 3, 7, and 9:

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

Here is an invalid sequence, since he never learned how to cook his third meal:

Day	1	2	3	4
Meal	1	2	1	4

Note that it is possible for Mr. Light to learn new meals in consecutive days (check the second sample test).

You will be given the meals he cooked on his first n days, except that some of them will be missing. Can you figure out the minimum number of meals he could have learned in those n days? Also output a valid sequence with the missing numbers filled with a valid meal he could have cooked that day.

Input

The first line of input contains a single integer T ($1 \leq T \leq 4096$), the number of test cases.

The first line of each test case contains a single integer n ($2 \leq n \leq 10^5$), the number of days.

The next line contains n space-separated integers v_1, v_2, ..., v_n ($0 \leq v_i \leq n$), where v_i is equal to the meal number he ate on the i^{th} day if $v_i > 0$. Otherwise if $v_i = 0$, the meal number is missing on that day. It is guaranteed that $v_1 = 1$, and there's at least one missing number.

It is guaranteed that for each test case, there is at least one way to fill the missing numbers to make a valid sequence.

The sum of n over all test cases doesn't exceed 6×10^6.

Output

For each test case, output the minimum number of meals k Mr. Light could have learned, on the first line.

One the second line, output n space-separated integers d_1, d_2, ..., d_n ($1 \leq d_i \leq k$), where d_i is a valid meal he could have cooked on the i^{th} day. The non-zero numbers from input must not change.

If there's more than one valid way to fill the missing meals, print any of them.

Example

meals.in	Standard Output
2 4 1 1 0 3 9 1 0 1 0 0 0 2 0 5	3 1 1 2 3 5 1 2 1 3 4 1 2 3 5