2018-2019 ICPC, NEERC, Southern Subregional Contest (Online Mirror, ACM-ICPC Rules, Teams Preferred)

A. Find a Number

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two positive integers d and s. Find minimal positive integer n which is divisible by d and has sum of digits equal to s.

Input
The first line contains two positive integers d and s ($1 \le d \le 500, 1 \le s \le 5000$) separated by space.

Output
Print the required number or -1 if it doesn't exist.

Examples

input
13 50
output
699998

input
61 2
output
1000000000000000000000000000001

input
15 50
output
-1

B. Berkomnadzor

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Berkomnadzor — Federal Service for Supervision of Communications, Information Technology and Mass Media — is a Berland federal executive body that protects ordinary residents of Berland from the threats of modern internet.

Berkomnadzor maintains a list of prohibited IPv4 subnets (blacklist) and a list of allowed IPv4 subnets (whitelist). All Internet Service Providers (ISPs) in Berland must configure the network equipment to block access to all IPv4 addresses matching the blacklist. Also ISPs must provide access (that is, do not block) to all IPv4 addresses matching the whitelist. If an IPv4 address does not match either of those lists, it's up to the ISP to decide whether to block it or not. An IPv4 address matches the blacklist (whitelist) if and only if it matches some subnet from the blacklist (whitelist). An IPv4 address can belong to a whitelist and to a blacklist at the same time, this situation leads to a contradiction (see no solution case in the output description).

An IPv4 address is a 32-bit unsigned integer written in the form $a.b.c.d$, where each of the values a, b, c, d is called an octet and is an integer from 0 to 255 written in decimal notation. For example, IPv4 address $192.168.0.1$ can be converted to a 32-bit number using the following expression $192 \cdot 2^{24} + 168 \cdot 2^{16} + 0 \cdot 2^8 + 1 \cdot 2^0$. First octet a encodes the most significant (leftmost) 8 bits, the octets b and c — the following blocks of 8 bits (in this order), and the octet d encodes the least significant (rightmost) 8 bits.

The IPv4 network in Berland is slightly different from the rest of the world. There are no reserved or internal addresses in Berland and use all 2^{32} possible values.

An IPv4 subnet is represented either as $a.b.c.d$ or as $a.b.c.d/x$ (where $0 \le x \le 32$). A subnet $a.b.c.d$ contains a single address $a.b.c.d$. A subnet $a.b.c.d/x$ contains all IPv4 addresses with x leftmost (most significant) bits equal to x leftmost bits of the address $a.b.c.d$. It is required that $32 - x$ rightmost (least significant) bits of subnet $a.b.c.d/x$ are zeroes.

Naturally it happens that all addresses matching subnet $a.b.c.d/x$ form a continuous range. The range starts with address $a.b.c.d$ (its rightmost $32 - x$ bits are zeroes). The range ends with address which x leftmost bits equal to x leftmost bits of address $a.b.c.d$

, and its $32 - x$ rightmost bits are all ones. Subnet contains exactly 2^{32-x} addresses. Subnet $a.b.c.d/32$ contains exactly one address and can also be represented by just $a.b.c.d$.

For example subnet $192.168.0.0/24$ contains range of 256 addresses. $192.168.0.0$ is the first address of the range, and $192.168.0.255$ is the last one.

Berkomnadzor's engineers have devised a plan to improve performance of Berland's global network. Instead of maintaining both whitelist and blacklist they want to build only a single optimised blacklist containing minimal number of subnets. The idea is to block all IPv4 addresses matching the optimised blacklist and allow all the rest addresses. Of course, IPv4 addresses from the old blacklist must remain blocked and all IPv4 addresses from the old whitelist must still be allowed. Those IPv4 addresses which matched neither the old blacklist nor the old whitelist may be either blocked or allowed regardless of their accessibility before.

Please write a program which takes blacklist and whitelist as input and produces optimised blacklist. The optimised blacklist must contain the minimal possible number of subnets and satisfy all IPv4 addresses accessibility requirements mentioned above.

IPv4 subnets in the source lists may intersect arbitrarily. Please output a single number `-1` if some IPv4 address matches both source whitelist and blacklist.

Input

The first line of the input contains single integer n ($1 \le n \le 2 \cdot 10^5$) — total number of IPv4 subnets in the input.

The following n lines contain IPv4 subnets. Each line starts with either '-' or '+' sign, which indicates if the subnet belongs to the blacklist or to the whitelist correspondingly. It is followed, without any spaces, by the IPv4 subnet in $a.b.c.d$ or $a.b.c.d/x$ format ($0 \le x \le 32$). The blacklist always contains at least one subnet.

All of the IPv4 subnets given in the input are valid. Integer numbers do not start with extra leading zeroes. The provided IPv4 subnets can intersect arbitrarily.

Output

Output `-1`, if there is an IPv4 address that matches both the whitelist and the blacklist. Otherwise output t — the length of the optimised blacklist, followed by t subnets, with each subnet on a new line. Subnets may be printed in arbitrary order. All addresses matching the source blacklist must match the optimised blacklist. All addresses matching the source whitelist must not match the optimised blacklist. You can print a subnet $a.b.c.d/32$ in any of two ways: as $a.b.c.d/32$ or as $a.b.c.d$.

If there is more than one solution, output any.

Examples

input
1 -149.154.167.99
output
1 0.0.0.0/0

input
4 -149.154.167.99 +149.154.167.100/30 +149.154.167.128/25 -149.154.167.120/29
output
2 149.154.167.99 149.154.167.120/29

input
5 -127.0.0.4/31 +127.0.0.8 +127.0.0.0/30 -195.82.146.208/29 -127.0.0.6/31
output
2 195.0.0.0/8 127.0.0.4/30

input
2 +127.0.0.1/32 -127.0.0.1
output
-1

C. Cloud Computing

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Buber is a Berland technology company that specializes in waste of investor's money. Recently Buber decided to transfer its infrastructure to a cloud. The company decided to rent CPU cores in the cloud for n consecutive days, which are numbered from 1 to n. Buber requires k CPU cores each day.

The cloud provider offers m tariff plans, the i-th tariff plan is characterized by the following parameters:

- l_i and r_i — the i-th tariff plan is available only on days from l_i to r_i, inclusive,
- c_i — the number of cores per day available for rent on the i-th tariff plan,
- p_i — the price of renting one core per day on the i-th tariff plan.

Buber can arbitrarily share its computing core needs between the tariff plans. Every day Buber can rent an arbitrary number of cores (from 0 to c_i) on each of the available plans. The number of rented cores on a tariff plan can vary arbitrarily from day to day.

Find the minimum amount of money that Buber will pay for its work for n days from 1 to n. If on a day the total number of cores for all available tariff plans is strictly less than k, then this day Buber will have to work on fewer cores (and it rents all the available cores), otherwise Buber rents exactly k cores this day.

Input

The first line of the input contains three integers n, k and m ($1 \le n, k \le 10^6, 1 \le m \le 2 \cdot 10^5$) — the number of days to analyze, the desired daily number of cores, the number of tariff plans.

The following m lines contain descriptions of tariff plans, one description per line. Each line contains four integers l_i, r_i, c_i, p_i ($1 \le l_i \le r_i \le n$, $1 \le c_i, p_i \le 10^6$), where l_i and r_i are starting and finishing days of the i-th tariff plan, c_i — number of cores, p_i — price of a single core for daily rent on the i-th tariff plan.

Output

Print a single integer number — the minimal amount of money that Buber will pay.

Examples

input
5 7 3 1 4 5 3 1 3 5 2 2 5 10 1
output
44

input
7 13 5 2 3 10 7 3 5 10 10 1 2 10 6 4 5 10 9 3 4 10 8
output
462

input
4 100 3 3 3 2 5 1 1 3 2 2 4 4 4
output
64

D. Garbage Disposal

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Enough is enough. Too many times it happened that Vasya forgot to dispose of garbage and his apartment stank afterwards. Now he wants to create a garbage disposal plan and stick to it.

For each of next n days Vasya knows a_i — number of *units of garbage* he will produce on the i-th day. Each *unit of garbage* must be disposed of either on the day it was produced or on the next day. Vasya disposes of garbage by putting it inside a bag and dropping

the bag into a garbage container. Each bag can contain up to k *units of garbage*. It is allowed to compose and drop multiple bags into a garbage container in a single day.

Being economical, Vasya wants to use as few bags as possible. You are to compute the minimum number of bags Vasya needs to dispose of all of his garbage for the given n days. No garbage should be left after the n-th day.

Input

The first line of the input contains two integers n and k ($1 \le n \le 2 \cdot 10^5, 1 \le k \le 10^9$) — number of days to consider and bag's capacity. The second line contains n space separated integers a_i ($0 \le a_i \le 10^9$) — the number of *units of garbage* produced on the i-th day.

Output

Output a single integer — the minimum number of bags Vasya needs to dispose of all garbage. Each *unit of garbage* should be disposed on the day it was produced or on the next day. No garbage can be left after the n-th day. In a day it is allowed to compose and drop multiple bags.

Examples

input
3 2 3 2 1
output
3

input
5 1 1000000000 1000000000 1000000000 1000000000 1000000000
output
5000000000

input
3 2 1 0 1
output
2

input
4 4 2 8 4 1
output
4

E. Getting Deals Done

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has a lot of work to do. Recently he has learned a new time management rule: "if a task takes five minutes or less, do it immediately". Polycarp likes the new rule, however he is not sure that five minutes is the optimal value. He supposes that this value d should be chosen based on existing task list.

Polycarp has a list of n tasks to complete. The i-th task has difficulty p_i, i.e. it requires exactly p_i minutes to be done. Polycarp reads the tasks one by one from the first to the n-th. If a task difficulty is d or less, Polycarp starts the work on the task immediately. If a task difficulty is strictly greater than d, he will not do the task at all. It is not allowed to rearrange tasks in the list. Polycarp doesn't spend any time for reading a task or skipping it.

Polycarp has t minutes in total to complete maximum number of tasks. But he does not want to work all the time. He decides to make a break after each group of m consecutive tasks he was working on. The break should take the same amount of time as it was spent in total on completion of these m tasks.

For example, if $n = 7$, $p = [3, 1, 4, 1, 5, 9, 2]$, $d = 3$ and $m = 2$ Polycarp works by the following schedule:

- Polycarp reads the first task, its difficulty is not greater than d ($p_1 = 3 \le d = 3$) and works for 3 minutes (i.e. the minutes 1, 2, 3);
- Polycarp reads the second task, its difficulty is not greater than d ($p_2 = 1 \le d = 3$) and works for 1 minute (i.e. the minute 4);
- Polycarp notices that he has finished $m = 2$ tasks and takes a break for $3 + 1 = 4$ minutes (i.e. on the minutes $5, 6, 7, 8$);
- Polycarp reads the third task, its difficulty is greater than d ($p_3 = 4 > d = 3$) and skips it without spending any time;
- Polycarp reads the fourth task, its difficulty is not greater than d ($p_4 = 1 \le d = 3$) and works for 1 minute (i.e. the minute 9);
- Polycarp reads the tasks 5 and 6, skips both of them ($p_5 > d$ and $p_6 > d$);

- Polycarp reads the 7-th task, its difficulty is not greater than d ($p_7 = 2 \le d = 3$) and works for 2 minutes (i.e. the minutes 10, 11);
- Polycarp notices that he has finished $m = 2$ tasks and takes a break for $1 + 2 = 3$ minutes (i.e. on the minutes $12, 13, 14$).

Polycarp stops exactly after t minutes. If Polycarp started a task but has not finished it by that time, the task is not considered as completed. It is allowed to complete less than m tasks in the last group. Also Polycarp considers acceptable to have shorter break than needed after the last group of tasks or even not to have this break at all — his working day is over and he will have enough time to rest anyway.

Please help Polycarp to find such value d, which would allow him to complete maximum possible number of tasks in t minutes.

Input

The first line of the input contains single integer c ($1 \le c \le 5 \cdot 10^4$) — number of test cases. Then description of c test cases follows. Solve test cases separately, test cases are completely independent and do not affect each other.

Each test case is described by two lines. The first of these lines contains three space-separated integers n, m and t ($1 \le n \le 2 \cdot 10^5, 1 \le m \le 2 \cdot 10^5, 1 \le t \le 4 \cdot 10^{10}$) — the number of tasks in Polycarp's list, the number of tasks he can do without a break and the total amount of time Polycarp can work on tasks. The second line of the test case contains n space separated integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 2 \cdot 10^5$) — difficulties of the tasks.

The sum of values n for all test cases in the input does not exceed $2 \cdot 10^5$.

Output

Print c lines, each line should contain answer for the corresponding test case — the maximum possible number of tasks Polycarp can complete and the integer value d ($1 \le d \le t$) Polycarp should use in time management rule, separated by space. If there are several possible values d for a test case, output any of them.

Examples

input
4 5 2 16 5 6 1 4 7 5 3 30 5 6 1 4 7 6 4 15 12 5 15 7 20 17 1 1 50 100
output
3 5 4 7 2 10 0 25

input
3 11 1 29 6 4 3 7 5 3 4 7 3 5 3 7 1 5 1 1 1 1 1 1 1 5 2 18 2 3 3 7 5
output
4 3 3 1 4 5

Note

In the first test case of the first example $n = 5$, $m = 2$ and $t = 16$. The sequence of difficulties is $[5, 6, 1, 4, 7]$. If Polycarp chooses $d = 5$ then he will complete 3 tasks. Polycarp will work by the following schedule:

- Polycarp reads the first task, its difficulty is not greater than d ($p_1 = 5 \le d = 5$) and works for 5 minutes (i.e. the minutes $1, 2, \ldots, 5$);
- Polycarp reads the second task, its difficulty is greater than d ($p_2 = 6 > d = 5$) and skips it without spending any time;
- Polycarp reads the third task, its difficulty is not greater than d ($p_3 = 1 \le d = 5$) and works for 1 minute (i.e. the minute 6);
- Polycarp notices that he has finished $m = 2$ tasks and takes a break for $5 + 1 = 6$ minutes (i.e. on the minutes $7, 8, \ldots, 12$);
- Polycarp reads the fourth task, its difficulty is not greater than d ($p_4 = 4 \le d = 5$) and works for 4 minutes (i.e. the minutes $13, 14, 15, 16$);
- Polycarp stops work because of $t = 16$.

In total in the first test case Polycarp will complete 3 tasks for $d = 5$. He can't choose other value for d to increase the number of completed tasks.

F. Debate

time limit per test: 3 seconds

memory limit per test: 256 megabytes
input: standard input
output: standard output

Elections in Berland are coming. There are only two candidates — Alice and Bob.

The main Berland TV channel plans to show political debates. There are n people who want to take part in the debate as a spectator. Each person is described by their influence and political views. There are four kinds of political views:

- supporting none of candidates (this kind is denoted as "00"),
- supporting Alice but not Bob (this kind is denoted as "10"),
- supporting Bob but not Alice (this kind is denoted as "01"),
- supporting both candidates (this kind is denoted as "11").

The direction of the TV channel wants to invite some of these people to the debate. The set of invited spectators should satisfy three conditions:

- at least half of spectators support Alice (i.e. $2 \cdot a \geq m$, where a is number of spectators supporting Alice and m is the total number of spectators),
- at least half of spectators support Bob (i.e. $2 \cdot b \geq m$, where b is number of spectators supporting Bob and m is the total number of spectators),
- the total influence of spectators is maximal possible.

Help the TV channel direction to select such non-empty set of spectators, or tell that this is impossible.

Input

The first line contains integer n ($1 \leq n \leq 4 \cdot 10^5$) — the number of people who want to take part in the debate as a spectator.

These people are described on the next n lines. Each line describes a single person and contains the string s_i and integer a_i separated by space ($1 \leq a_i \leq 5000$), where s_i denotes person's political views (possible values — "00", "10", "01", "11") and a_i — the influence of the i-th person.

Output

Print a single integer — maximal possible total influence of a set of spectators so that at least half of them support Alice and at least half of them support Bob. If it is impossible print 0 instead.

Examples

input
6 11 6 10 4 01 3 00 3 00 7 00 9
output
22

input
5 11 1 01 1 00 100 10 1 01 1
output
103

input
6 11 19 10 22 00 18 00 29 11 29 10 28
output
105

input
3 00 5000 00 5000 00 5000
output

```
0
```

Note

In the first example 4 spectators can be invited to maximize total influence: 1, 2, 3 and 6. Their political views are: "11", "10", "01" and "00". So in total 2 out of 4 spectators support Alice and 2 out of 4 spectators support Bob. The total influence is $6 + 4 + 3 + 9 = 22$.

In the second example the direction can select all the people except the 5-th person.

In the third example the direction can select people with indices: 1, 4, 5 and 6.

In the fourth example it is impossible to select any non-empty set of spectators.

G. Monsters and Potions

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp is an introvert person. In fact he is so much of an introvert that he plays "Monsters and Potions" board game alone. The board of the game is a row of n cells. The cells are numbered from 1 to n from left to right. There are three types of cells: a cell containing a single monster, a cell containing a single potion or a *blank* cell (it contains neither a monster nor a potion).

Polycarp has m tokens representing heroes fighting monsters, which are initially located in the blank cells $s_1, s_2, \ldots, s_m$. Polycarp's task is to choose a single cell (rally point) and one by one move all the heroes into this cell. A rally point can be a cell of any of three types.

After Policarp selects a rally point, he picks a hero and orders him to move directly to the point. Once that hero reaches the point, Polycarp picks another hero and orders him also to go to the point. And so forth, until all the heroes reach the rally point cell. While going to the point, a hero can not deviate from the direct route or take a step back. A hero just moves cell by cell in the direction of the point until he reaches it. It is possible that multiple heroes are simultaneously in the same cell.

Initially the i-th hero has h_i hit points (HP). Monsters also have HP, different monsters might have different HP. And potions also have HP, different potions might have different HP.

If a hero steps into a cell which is blank (i.e. doesn't contain a monster/potion), hero's HP does not change.

If a hero steps into a cell containing a monster, then the hero and the monster fight. If monster's HP is strictly higher than hero's HP, then the monster wins and Polycarp loses the whole game. If hero's HP is greater or equal to monster's HP, then the hero wins and monster's HP is subtracted from hero's HP. I.e. the hero survives if his HP drops to zero, but dies (and Polycarp looses) if his HP becomes negative due to a fight. If a hero wins a fight with a monster, then the monster disappears, and the cell becomes blank.

If a hero steps into a cell containing a potion, then the hero drinks the potion immediately. As a result, potion's HP is added to hero's HP, the potion disappears, and the cell becomes blank.

Obviously, Polycarp wants to win the game. It means that he must choose such rally point and the order in which heroes move, that every hero reaches the rally point and survives. I.e. Polycarp loses if a hero reaches rally point but is killed by a monster at the same time. Polycarp can use any of n cells as a rally point — initially it can contain a monster, a potion, or be a blank cell with or without a hero in it.

Help Polycarp write a program to choose a rally point and the order in which heroes move.

Input

The first line of the input contains two integers n and m ($1 \le n \le 100$; $1 \le m \le n$) — length of the game board and the number of heroes on it.

The following m lines describe heroes. Each line contains two integers s_i and h_i ($1 \le s_i \le n$; $1 \le h_i \le 10^6$), where s_i is the initial position and h_i is the initial HP of the i-th hero. It is guaranteed that each cell s_i is blank. It is also guaranteed that all s_i are different.

The following line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^6 \le a_j \le 10^6$), where a_j describes the i-th cell of the game board:

- $a_j = 0$ means that the i-th cell is blank,
- $a_j < 0$ means that the i-th cell contains monster with positive HP of $-a_j$,
- $a_j > 0$ means that the i-th cell contains potion with a_j HP.

Output

On the first line of the output print the index of the rally point cell.

On the second line print m integers — the order in which heroes should move to the rally point. Heroes are numbered from 1 to m in the order they are given in the input.

If there are multiple solutions, print any of them.

If it is impossible to find a rally point which can be reached by all heroes, print a single integer -1 in the output.

Examples

input
8 3 8 2 1 3 4 9 0 3 -5 0 -5 -4 -1 0
output
6 3 1 2

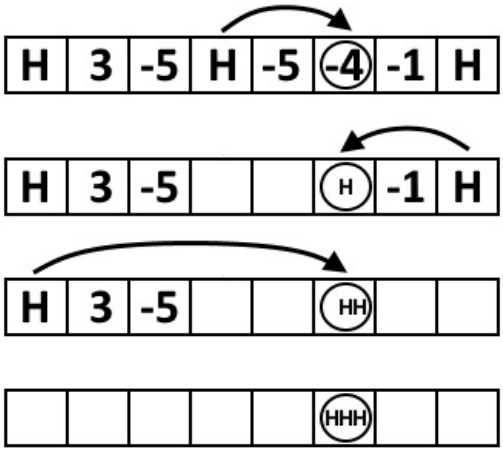
input
1 1 1 1 0
output
1 1

input
3 2 1 1 3 1 0 -5000 0
output
-1

input
8 3 1 15 5 10 8 1 0 -5 -5 -5 0 -5 -5 0
output
7 2 1 3

Note

The picture illustrates the first example:



H. BerOS File Suggestion

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp is working on a new operating system called BerOS. He asks you to help with implementation of a file suggestion feature.

There are n files on hard drive and their names are $f_1, f_2, \ldots, f_n$. Any file name contains between 1 and 8 characters, inclusive. All file names are unique.

The file suggestion feature handles queries, each represented by a string s. For each query s it should count number of files containing s as a substring (i.e. some continuous segment of characters in a file name equals s) and suggest any such file name.

For example, if file names are "`read.me`", "`hosts`", "`ops`", and "`beros.18`", and the query is "`os`", the number of matched files is 2 (two file names contain "`os`" as a substring) and suggested file name can be either "`hosts`" or "`beros.18`".

Input
The first line of the input contains integer n ($1 \le n \le 10000$) — the total number of files.

The following n lines contain file names, one per line. The i-th line contains f_i — the name of the i-th file. Each file name contains between 1 and 8 characters, inclusive. File names contain only lowercase Latin letters, digits and dot characters ('.'). Any sequence of valid characters can be a file name (for example, in BerOS ".", ".." and "..." are valid file names). All file names are unique.

The following line contains integer q ($1 \le q \le 50000$) — the total number of queries.

The following q lines contain queries $s_1, s_2, \ldots, s_q$, one per line. Each s_j has length between 1 and 8 characters, inclusive. It contains only lowercase Latin letters, digits and dot characters ('.').

Output
Print q lines, one per query. The j-th line should contain the response on the j-th query — two values c_j and t_j, where

- c_j is the number of matched files for the j-th query,
- t_j is the name of any file matched by the j-th query. If there is no such file, print a single character '-' instead. If there are multiple matched files, print any.

Example

input
4 test contests test. .test 6 ts . st. .test contes. st
output
1 contests 2 .test 1 test. 1 .test 0 - 4 test.

I. Privatization of Roads in Berland

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n cities and m two-way roads in Berland, each road connecting two distinct cities.

Recently the Berland government has made a tough decision to transfer ownership of the roads to private companies. In total, there are 100500 private companies in Berland, numbered by integers from 1 to 100500. After the privatization, every road should belong to exactly one company.

The anti-monopoly committee demands that after the privatization each company can own at most two roads. The urbanists of Berland also stated their opinion: each city should be adjacent to the roads owned by at most k companies.

Help the government to distribute the roads between the companies so that both conditions are satisfied. That is, each company gets at most two roads, and each city has roads of at most k distinct companies adjacent to it.

Input
Input contains one or several test cases. The first line contains an integer t ($1 \le t \le 300$) — the number of test cases in the input. Solve test cases separately, test cases are completely independent and do not affect each other.

The following lines describe the test cases. Each case starts with a line consisting of three space-separated integers n, m and k ($2 \le n \le 600$, $1 \le m \le 600$, $1 \le k \le n - 1$) — the number of cities, the number of roads and the maximum diversity of the roads adjacent to a city.

Then m lines follow, each having a pair of space-separated integers a_i, b_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$). It means that the i-th road connects cities a_i and b_i. All roads are two-way. There is at most one road between a pair of the cities.

The sum of n values for all test cases doesn't exceed 600. The sum of m values for all test cases doesn't exceed 600.

Output
Print t lines: the i-th line should contain the answer for the i-th test case. For a test case, print a sequence of integers $c_1, c_2, \ldots, c_m$ separated by space, where c_i ($1 \le c_i \le 100500$) is the company which owns the i-th road in your plan. If there are multiple solutions, output any of them. If there is no solution for a test case, print $c_1 = c_2 = \ldots = c_m = 0$.

Example

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

output
1 2 3 2 1 1 2 3 0 0 0 0 0 0

J. Streets and Avenues in Berhattan

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Berhattan is the capital of Berland. There are n streets running parallel in the east-west direction (horizontally), and there are m avenues running parallel in the south-north direction (vertically). Each street intersects with each avenue, forming a crossroad. So in total there are $n \cdot m$ crossroads in Berhattan.

Recently, the government has changed in Berland. The new government wants to name all avenues and all streets after heroes of revolution.

The special committee prepared the list of k names. Only these names can be used as new names for streets and avenues. Each name can be used at most once.

The members of committee want to name streets and avenues in the way that minimizes inconvenience for residents. They believe that if street and avenue names start with the same letter, then their crossroad will be *inconvenient*. Hence only the first letter of each name matters.

Given first letters of k names, find C — minimal possible number of inconvenient crossroads in Berhattan after the naming process.

Input

Input contains one or several test cases to process. The first line contains t ($1 \le t \le 30000$) — the number of test cases. Solve test cases separately, test cases are completely independent and do not affect each other.

The description of t test cases follows. Each test case starts with line with space-separated numbers n, m, k ($1 \le n, m \le 30000$; $n + m \le k \le 2 \cdot 10^5$) — the number of streets, number of avenues and the number of names in the committee's list, respectively.

The the second line of each test case contains a string of k uppercase English letters. i-th letter of the string is the first letter of i-th name from the committee's list.

It's guaranteed that the sum of numbers n from all test cases is not greater than 30000. Similarly, the sum of numbers m from all test cases is not greater than 30000. The sum of numbers k from all test cases is not greater than $2 \cdot 10^5$.

Output

For each test case print single number C in the separate line — minimal possible number of inconvenient crossroads in Berhattan after the naming process.

Examples

input
2 2 3 9 EEZZEEZZZ 2 7 9 EEZZEEZZZ

output
0 4

input
2 4 4 8

CZBBCZBC 1 1 4 TTCT
output
1 0

K. Video Posts

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp took n videos, the duration of the i-th video is a_i seconds. The videos are listed in the chronological order, i.e. the 1-st video is the earliest, the 2-nd video is the next, ..., the n-th video is the last.

Now Polycarp wants to publish exactly k ($1 \le k \le n$) posts in Instabram. Each video should be a part of a single post. The posts should preserve the chronological order, it means that the first post should contain one or more of the earliest videos, the second post should contain a block (one or more videos) going next and so on. In other words, if the number of videos in the j-th post is s_j then:

- $s_1 + s_2 + \dots + s_k = n$ ($s_i > 0$),
- the first post contains the videos: $1, 2, \dots, s_1$;
- the second post contains the videos: $s_1 + 1, s_1 + 2, \dots, s_1 + s_2$;
- the third post contains the videos: $s_1 + s_2 + 1, s_1 + s_2 + 2, \dots, s_1 + s_2 + s_3$;
- ...
- the k-th post contains videos: $n - s_k + 1, n - s_k + 2, \dots, n$.

Polycarp is a perfectionist, he wants the total duration of videos in each post to be the same.

Help Polycarp to find such positive integer values $s_1, s_2, \dots, s_k$ that satisfy all the conditions above.

Input

The first line contains two integers n and k ($1 \le k \le n \le 10^5$). The next line contains n positive integer numbers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^4$), where a_i is the duration of the i-th video.

Output

If solution exists, print "Yes" in the first line. Print k positive integers $s_1, s_2, \dots, s_k$ ($s_1 + s_2 + \dots + s_k = n$) in the second line. The total duration of videos in each post should be the same. It can be easily proven that the answer is unique (if it exists).

If there is no solution, print a single line "No".

Examples

input
6 3 3 3 1 4 1 6
output
Yes 2 3 1

input
3 3 1 1 1
output
Yes 1 1 1

input
3 3 1 1 2
output
No

input
3 1 1 10 100
output
Yes 3

L. Odd Federalization

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Berland has n cities, some of which are connected by roads. Each road is bidirectional, connects two distinct cities and for each two cities there's at most one road connecting them.

The president of Berland decided to split country into r states in such a way that each city will belong to exactly one of these r states.

After this split each road will connect either cities of the same state or cities of the different states. Let's call roads that connect two cities of the same state "*inner*" roads.

The president doesn't like odd people, odd cities and odd numbers, so he wants this split to be done in such a way that each city would have even number of "*inner*" roads connected to it.

Please help president to find smallest possible r for which such a split exists.

Input

The input contains one or several test cases. The first input line contains a single integer number t — number of test cases. Then, t test cases follow. Solve test cases separately, test cases are completely independent and do not affect each other.

Then t blocks of input data follow. Each block starts from empty line which separates it from the remaining input data. The second line of each block contains two space-separated integers n, m ($1 \le n \le 2000$, $0 \le m \le 10000$) — the number of cities and number of roads in the Berland. Each of the next m lines contains two space-separated integers — x_i, y_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$), which denotes that the i-th road connects cities x_i and y_i. Each pair of cities are connected by at most one road.

Sum of values n across all test cases doesn't exceed 2000. Sum of values m across all test cases doesn't exceed 10000.

Output

For each test case first print a line containing a single integer r — smallest possible number of states for which required split is possible. In the next line print n space-separated integers in range from 1 to r, inclusive, where the j-th number denotes number of state for the j-th city. If there are multiple solutions, print any.

Example

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

M. Algoland and Berland

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Once upon a time Algoland and Berland were a single country, but those times are long gone. Now they are two different countries, but their cities are scattered on a common territory.

All cities are represented as points on the Cartesian plane. Algoland consists of a cities numbered from 1 to a. The coordinates of the i-th Algoland city are a pair of integer numbers (xa_i, ya_i). Similarly, Berland consists of b cities numbered from 1 to b. The coordinates of the j-th Berland city are a pair of integer numbers (xb_j, yb_j). *No three of the $a + b$ mentioned cities lie on a single straight line*.

As the first step to unite the countries, Berland decided to build several bidirectional freeways. Each freeway is going to be a line segment that starts in a Berland city and ends in an Algoland city. Freeways can't intersect with each other at any point except freeway's start or end. Moreover, the freeways have to connect all $a + b$ cities. Here, connectivity means that one can get from any of the specified $a + b$ cities to any other of the $a + b$ cities using freeways. Note that all freeways are bidirectional, which means that one can drive on each of them in both directions.

Mayor of each of the b Berland cities allocated a budget to build the freeways that start from this city. Thus, you are given the numbers $r_1, r_2, \ldots, r_b$, where r_j is the number of freeways that are going to start in the j-th Berland city. The total allocated budget is very tight and only covers building the minimal necessary set of freeways. In other words, $r_1 + r_2 + \cdots + r_b = a + b - 1$.

Help Berland to build the freeways so that:

- each freeway is a line segment connecting a Berland city and an Algoland city,
- no freeways intersect with each other except for the freeway's start or end,
- freeways connect all $a + b$ cities (all freeways are bidirectional),
- there are r_j freeways that start from the j-th Berland city.

Input

Input contains one or several test cases. The first input line contains a single integer number t ($1 \le t \le 3000$) — number of test cases. Then, t test cases follow. Solve test cases separately, test cases are completely independent and do not affect each other.

Each test case starts with a line containing space-separated integers a and b ($1 \le a, b \le 3000$) — numbers of Algoland cities and number of Berland cities correspondingly.

The next line contains b space-separated integers $r_1, r_2, \ldots, r_b$ ($1 \le r_b \le a$) where r_j is the number of freeways, that should start in the j-th Berland city. It is guaranteed that $r_1 + r_2 + \cdots + r_b = a + b - 1$.

The next a lines contain coordinates of the Algoland cities — pairs of space-separated integers xa_i, ya_i ($-10000 \le xa_i, ya_i \le 10000$). The next b lines contain coordinates of the Berland cities — pairs of space-separated integers xb_i, yb_i ($-10000 \le xb_i, yb_i \le 10000$). All cities are located at distinct points, no three of the $a + b$ cities lie on a single straight line.

Sum of values a across all test cases doesn't exceed 3000. Sum of values b across all test cases doesn't exceed 3000.

Output

For each of the t test cases, first print "YES" if there is an answer or "NO" otherwise.

If there is an answer, print the freeway building plan in the next $a + b - 1$ lines. Each line of the plan should contain two space-separated integers j and i which means that a freeway from the j-th Berland city to the i-th Algoland city should be built. If there are multiple solutions, print any.

Example

input
2 2 3 1 1 2 0 0 1 1 1 2 3 2 4 0 1 1 1 0 0 0 1
output
YES 2 2 1 2 3 2 3 1 YES 1 1