Problem A. Award Ceremony

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

All-Berland programming contest comes to an end. In total, n teams participated in it. Like in ACM-ICPC, current results stopped refreshing one hour before the contest ends. So at the Award Ceremony, results are partially known. For each team the value a_i is given — the number of points the i-th team has earned before the last hour of the contest. Besides that, the Jury has evaluated all submissions sent during the last hour and knows values d_i — the number of points earned by the i-th team during the last hour (these values can be negative, which means that a team can lose points).

Before the contest, each team got unique id from 1 to n. According to the contest rules, a team with more points takes a higher place. If two or more teams have equal number of points, the team with lower id will take the higher place. So no two teams can share the same place.

The Award Ceremony proceeds in the following way. At the beginning of the ceremony, a large screen shows the results for the time moment "one hour before the end", which means that the i-th team has a_i points. Then the Jury unfreezes results of the teams one by one in some order. When result of the j-th team is unfrozen, its score changes from a_j to $a_j + d_j$. At this time the table of results is modified and the place of the team can change. The unfreezing of the j-th team is followed by the applause from the audience with duration of $|x_j - y_j|$ seconds, where x_j is the place of the j-th team before unfreezing and y_j is the place right after the unfreezing. For example, if the team does not change the place, there is no applause from the audience. As you can see, during the Award Ceremony, each team will be unfrozen exactly once.

Your task is to find such an order to unfreeze all the teams that the total duration of applause is maximum possible.

Input

The first line of the input file contains a single integer n ($1 \le n \le 100$) — the number of teams.

Each of the next n lines contains two integers a_i and d_i ($1 \le a_i \le 100$, $-100 \le d_i \le 100$) — the number of points the i-th team has earned before the last hour of the contest and the number of points earned by this team during the last hour. It is possible that after unfreezing a team will have a negative score.

Output

Print the only integer — maximal total applause duration in seconds if the Jury can choose any order of the teams to unfreeze.

Examples

standard input	standard output
4 17 -14 52 -5 1 52 6 0	4
5 4 5 3 2 5 -3 6 -2 4 3	14

Note

In the first example the initial standings are:

1. Team 2, 52 points
2. Team 1, 17 points
3. Team 4, 6 points
4. Team 3, 1 point

Here any order of unfreezing the teams leads to 4 seconds of applause in total. For example, let's unfreeze teams in their order from the Team 1 to the Team 4.

After the Team 1 became unfrozen the standings are:

1. Team 2, 52 points
2. Team 4, 6 points
3. Team 1, 3 points
4. Team 3, 1 point

So there is 1 second of applause, because the difference between old and new places $|2 - 3| = 1$.

After the Team 2 became unfrozen the standings are:

1. Team 2, 47 points
2. Team 4, 6 points
3. Team 1, 3 points
4. Team 3, 1 point

The place of the Team 2 has not changed, so no applause during unfreezing.

After the Team 3 became unfrozen the standings are:

1. Team 3, 53 point
2. Team 2, 47 points
3. Team 4, 6 points
4. Team 1, 3 points

The place of the Team 3 has changed from 4 to 1, so the duration of applause is $|4 - 1| = 3$.

The unfreezing of the Team 4 has not changed any place because $d_4 = 0$.

Therefore, the total duration of applause is $1 + 0 + 3 + 0 = 4$ seconds.

Problem B. Bottles

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

Nick has n bottles of soda left after his birthday. Each bottle is described by two values: remaining amount of soda a_i and bottle volume b_i ($a_i \le b_i$).

Nick has decided to pour all remaining soda into minimal number of bottles, moreover he has to do it as soon as possible. Nick spends x seconds to pour x units of soda from one bottle to another.

Nick asks you to help him to determine k — the minimal number of bottles to store all remaining soda and t — the minimal time to pour soda into k bottles. A bottle can't store more soda than its volume. All remaining soda should be saved.

Input

The first line contains positive integer n ($1 \le n \le 100$) — the number of bottles.

The second line contains n positive integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$), where a_i is the amount of soda remaining in the i-th bottle.

The third line contains n positive integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 100$), where b_i is the volume of the i-th bottle.

It is guaranteed that $a_i \le b_i$ for any i.

Output

The only line should contain two integers k and t, where k is the minimal number of bottles that can store all the soda and t is the minimal time to pour the soda into k bottles.

Examples

standard input	standard output
4 3 3 4 3 4 7 6 5	2 6
2 1 1 100 100	1 1
5 10 30 5 6 24 10 41 7 8 24	3 11

Note

In the first example Nick can pour soda from the first bottle to the second bottle. It will take 3 seconds. After it the second bottle will contain $3 + 3 = 6$ units of soda. Then he can pour soda from the fourth bottle to the second bottle and to the third bottle: one unit to the second and two units to the third. It will take $1 + 2 = 3$ seconds. So, all the soda will be in two bottles and he will spend $3 + 3 = 6$ seconds to do it.

Problem C. Running Over The Bridges

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

Polycarp is playing a game called "Running Over The Bridges". In this game he has to run over n bridges from the left to the right. Bridges are arranged one after the other, so the i-th bridge begins where the $(i-1)$-th bridge ends.

You have the following data about bridges: l_i and t_i — the length of the i-th bridge and the maximum allowed time which Polycarp can spend running over the i-th bridge. Thus, if Polycarp is in the beginning of the bridge i at the time T then he has to leave it at the time $T + t_i$ or earlier. It is allowed to reach the right end of a bridge exactly at the time $T + t_i$.

Polycarp can run from the left side to the right one with speed 0.5, so he will run over a bridge with length s in time $2 \cdot s$. Besides, he has several magical drinks. If he uses one drink, his speed increases twice (i.e. to value 1) for r seconds. All magical drinks are identical. Please note that Polycarp can use a drink only at **integer** moments of time, and he drinks it instantly and completely. Additionally, if Polycarp uses a drink at the moment T he can use the next drink not earlier than at the moment $T + r$.

What is the minimal number of drinks Polycarp has to use to run over all n bridges? If this number is not greater than 10^5, then you have to find out the moments of time when Polycarp has to use each magical drink.

Input

The first line contains two integers n and r ($1 \le n \le 2 \cdot 10^5$, $1 \le r \le 10^{12}$) — the number of bridges and the duration of the effect of a magical drink.

The second line contains a sequence of integers $l_1, l_2, \ldots, l_n$ ($1 \le l_i \le 5 \cdot 10^6$), where l_i is equal to the length of the i-th bridge.

The third line contains a sequence of integers $t_1, t_2, \ldots, t_n$ ($1 \le t_i \le 10^7$), where t_i is equal to the maximum allowed time which Polycarp can spend running over the i-th bridge.

Output

The first line of the output should contain k — the minimal number of drinks which Polycarp has to use, or `-1` if there is no solution.

If the solution exists and the value of k is not greater than 10^5 then output k integers on the next line — moments of time from beginning of the game when Polycarp has to use drinks. Print the moments of time in chronological order. If there are several solutions, you can output any of them.

Examples

standard input	standard output
1 3 7 10	2 0 3
3 3 3 3 3 3 3 2	-1
3 100000 5 5 5 5 7 8	1 0
4 1000 1 2 3 4 10 9 10 9	0

Note

In the first case, there is only one bridge and it is clear that Polycarp cannot run over it without magical drinks. So, if he will use one magical drink on start (moment of time 0), and the second one — three seconds later (moment of time 3), he will be able to reach the end of the bridge in time. Please note, in this case there are several possible answers to the problem. For example, Polycarp can use the first drink at the moment of time 4 and the second one — at the moment of time 7.

In the second case, Polycarp cannot run over all bridges even if he will use magical drinks. So, answer in this case is -1.

In the fourth case, Polycarp can run over all bridges without magical drinks.

Problem D. Olympiad in Programming and Sports

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

There are n students at Berland State University. Every student has two skills, each measured as a number: a_i — the programming skill and b_i — the sports skill.

It is announced that an Olympiad in programming and sports will be held soon. That's why Berland State University should choose two teams: one to take part in the programming track and one to take part in the sports track.

There should be exactly p students in the programming team and exactly s students in the sports team. A student can't be a member of both teams.

The university management considers that the strength of the university on the Olympiad is equal to the sum of two values: the programming team strength and the sports team strength. The strength of a team is the sum of skills of its members in the corresponding area, so the strength of the programming team is the sum of all a_i and the strength of the sports team is the sum of all b_i over corresponding team members.

Help Berland State University to compose two teams to maximize the total strength of the university on the Olympiad.

Input

The first line contains three positive integer numbers n, p and s ($2 \le n \le 3000$, $p + s \le n$) — the number of students, the size of the programming team and the size of the sports team.

The second line contains n positive integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 3000$), where a_i is the programming skill of the i-th student.

The third line contains n positive integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 3000$), where b_i is the sports skill of the i-th student.

Output

In the first line, print the the maximum strength of the university on the Olympiad. In the second line, print p numbers — the members of the programming team. In the third line, print s numbers — the members of the sports team.

The students are numbered from 1 to n as they are given in the input. All numbers printed in the second and in the third lines should be distinct and can be printed in arbitrary order.

If there are multiple solutions, print any of them.

Examples

standard input	standard output
5 2 2 1 3 4 5 2 5 3 2 1 4	18 3 4 1 5
4 2 2 10 8 8 3 10 7 9 4	31 1 2 3 4
5 3 1 5 2 5 1 7 6 3 1 6 3	23 1 3 5 4

Problem E. Bulmart

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	512 megabytes

A new trade empire is rising in Berland. Bulmart, an emerging trade giant, decided to dominate the market of ... shovels! And now almost every city in Berland has a Bulmart store, and some cities even have several of them! The only problem is, at the moment sales are ... let's say a little below estimates. Some people even say that shovels retail market is too small for such a big company to make a profit. But the company management believes in the future of that market and seeks new ways to increase income.

There are n cities in Berland connected with m bi-directional roads. All roads have equal lengths. It can happen that it is impossible to reach a city from another city using only roads. There is no road which connects a city to itself. Any pair of cities can be connected by at most one road.

There are w Bulmart stores in Berland. Each of them is described by three numbers:

- c_i — the number of city where the i-th store is located (a city can have no stores at all or have several of them),
- k_i — the number of shovels in the i-th store,
- p_i — the price of a single shovel in the i-th store (in burles).

The latest idea of the Bulmart management is to create a program which will help customers get shovels as fast as possible for affordable budget. Formally, the program has to find the minimum amount of time needed to deliver r_j shovels to the customer in the city g_j for the total cost of no more than a_j burles. The delivery time between any two adjacent cities is equal to 1. If shovels are delivered from several cities, the delivery time is equal to the arrival time of the last package. The delivery itself is free of charge.

The program needs to find answers to q such queries. Each query has to be processed independently from others, i.e. a query does not change number of shovels in stores for the next queries.

Input

The first line contains two integers n, m ($1 \le n \le 5000$, $0 \le m \le min(5000, n \cdot (n-1)/2)$). Each of the next m lines contains two integers x_e and y_e, meaning that the e-th road connects cities x_e and y_e ($1 \le x_e, y_e \le n$).

The next line contains a single integer w ($1 \le w \le 5000$) — the total number of Bulmart stores in Berland. Each of the next w lines contains three integers describing the i-th store: c_i, k_i, p_i ($1 \le c_i \le n, 1 \le k_i, p_i \le 2 \cdot 10^5$).

The next line contains a single integer q ($1 \le q \le 1000$) — the number of queries. Each of the next q lines contains three integers describing the j-th query: g_j, r_j and a_j ($1 \le g_j \le n$, $1 \le r_j, a_j \le 10^9$)

Output

Output q lines. On the j-th line, print an answer for the j-th query — the minimum amount of time needed to deliver r_j shovels to the customer in city g_j spending no more than a_j burles. Print `-1` if there is no solution for the j-th query.

Example

standard input	standard output
6 4 4 2 5 4 1 2 3 2 2 4 1 2 3 2 3 6 1 2 6 2 3 7 3 1 2 4 3 8 5 2 5 6 1 10	2 -1 2 2 3 -1

Problem F. Expression Queries

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

A *simplified arithmetic expression* (SAE) is an arithmetic expression defined by the following grammar:

- `<SAE> ::= <Number> | <SAE>+<SAE> | <SAE>*<SAE> | (<SAE>)`
- `<Number> ::= <Digit> | <Digit><Number>`
- `<Digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

In other words it's a correct arithmetic expression that is allowed to contain brackets, numbers (possibly with leading zeros), multiplications and additions. For example expressions "`(0+01)`", "`0`" and "`1*(0)`" are simplified arithmetic expressions, but expressions "`2-1`", "`+1`" and "`1+2)`" are not.

Given a string $s_1 s_2 ... s_{|s|}$ that represents a SAE; s_i denotes the i-th character of the string which can be either a digit ('`0`'-'`9`'), a plus sign ('`+`'), a multiplication sign ('`*`'), an opening round bracket '`(`' or a closing round bracket '`)`'.

A part $s_l s_{l+1} ... s_r$ of this string is called a *sub-expression* if and only if it is a SAE.

You task is to answer m queries, each of which is a pair of integers l_i, r_i ($1 \le l_i \le r_i \le |s|$). For each query determine whether the corresponding part of the given string is a sub-expression and in case it's a sub-expression calculate its value modulo 1000000007 ($10^9 + 7$). The values should be calculated using standard operator priorities.

Input

The first line of the input contains non-empty string s ($1 \le |s| \le 4 \cdot 10^5$) which represents a correct SAE. Each character of the string can be one of the following characters: '`*`', '`+`', '`(`', '`)`' or a digit ('`0`'-'`9`'). The expression might contain extra-huge numbers.

The second line contains an integer m ($1 \le m \le 4 \cdot 10^5$) which is the number of queries. Each of the next m lines contains two space-separated integers l_i, r_i ($1 \le l_i \le r_i \le |s|$) — the i-th query.

Output

The i-th number of output should be the answer for the i-th query. If the i-th query corresponds to a valid sub-expression output the value of the sub-expression modulo 1000000007 ($10^9 + 7$). Otherwise output -1 as an answer for the query. Print numbers on separate lines.

Examples

standard input	standard output
((1+2)*3+101*2) 6 8 14 1 6 2 10 11 14 5 5 4 5	205 -1 10 2 2 -1
(01) 1 1 4	1

Problem G. Delete Them

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

Polycarp is a beginner programmer. He is studying how to use a command line.

Polycarp faced the following problem. There are n files in a directory and he needs to delete some of them. Polycarp wants to run a single delete command with filename pattern as an argument. All the files to be deleted should match the pattern and all other files shouldn't match the pattern.

Polycarp doesn't know about an asterisk '`*`', the only special character he knows is a question mark '`?`' which matches any single character. All other characters in the pattern match themselves only.

Formally, a pattern matches a filename if and only if they have equal lengths and all characters in the corresponding positions are equal except when the character in the pattern is '`?`', in which case the corresponding filename character does not matter.

For example, the filename pattern "`a?ba?`":

- matches filenames "`aabaa`", "`abba.`", "`a.ba9`" and "`a.ba.`";
- does not match filenames "`aaba`", "`abaab`", "`aabaaa`" and "`aabaa.`".

Help Polycarp find a pattern which matches files to be deleted and only them or report if there is no such pattern.

Input

The first line of the input contains two integers n and m ($1 \le m \le n \le 100$) — the total number of files and the number of files to be deleted.

The following n lines contain filenames, single filename per line. All filenames are non-empty strings containing only lowercase English letters, digits and dots ('`.`'). The length of each filename doesn't exceed 100. It is guaranteed that all filenames are distinct.

The last line of the input contains m distinct integer numbers in ascending order $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$) — indices of files to be deleted. All files are indexed from 1 to n in order of their appearance in the input.

Output

If the required pattern exists, print "`Yes`" in the first line of the output. The second line should contain the required pattern. If there are multiple solutions, print any of them.

If the required pattern doesn't exist, print the only line containing "`No`".

Examples

standard input	standard output
3 2 ab ac cd 1 2	Yes a?
5 3 test tezt test. .est tes. 1 4 5	Yes ?es?
4 4 a b c dd 1 2 3 4	No
6 3 .svn .git 1 2 3	Yes .???

Problem H. Minimum and Maximum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

This is an interactive problem. You have to use `flush` *operation right after printing each line. For example, in C++ you should use function* `fflush(stdout)`*, in Java —* `System.out.flush()`*, in Pascal —* `flush(output)` *and in Python —* `sys.stdout.flush()`*.*

In this problem, you need to find maximal and minimal elements of an array. What could be simpler?

You can imagine that the jury has an array, and initially you know the only number n — array's length.

Array's elements are numbered from 1 to n. You are allowed to compare two elements of the array by using their indices i and j. There are three possible responses to this query: '`<`' (if a_i is less than a_j), '`=`' (if a_i is equal to a_j) and finally '`>`' (if a_i is greater than a_j).

It's known that it's always possible to find both maximal and minimal elements of the array by using no more than $f(n) = \lceil \frac{3 \cdot n}{2} \rceil - 2$ comparisons, where $\lceil x \rceil$ is the result of rounding x up.

Write the program that will find positions of the minimum and the maximum in the jury's array of length n, by using no more than $f(n)$ comparisons.

Interaction Protocol

Each test for this problem will contain one or more arrays. You have to find positions of minimal and maximal elements for each of these arrays. The first line of the input contains integer T ($1 \le T \le 1000$) — number of arrays in the test.

Thus, at the beginning, you program should read number T, and then it should solve the problem for T jury's arrays one by one.

Then input for each array goes. Firstly, your program has to read the number n ($1 \le n \le 50$) — the length of the array. It will be provided in the next line of the input.

Further, your program can perform comparisons or report that the answer is found.

- To perform a comparison, you have to output string of the following pattern «`? i j`» (i and j must be integer numbers from 1 to n) — the indices of the elements to compare in the current query.
- To report the indices of minimal and maximal elements of the hidden array, your program have to output a line in the form «`! i j`» (i and j must be integer numbers from 1 to n), where i is an index of the minimal element of array, and j is an index of the maximal element of the array. If there are several possible answers to the problem, you can output any of them.

There are several possible responses for a comparison:

- '`<`' — if a_i is less than a_j,
- '`=`' — if a_i is equal to a_j,
- '`>`' — if a_i is greater than a_j.

For an array of length n your program can make at most $f(n) = \lceil \frac{3 \cdot n}{2} \rceil - 2$ comparisons. Note that the operation of reporting an answer («`! i j`») is not included into the value of $f(n)$.

After the answer is reported, your program has to solve the problem for the next array or it should terminate if all T arrays are processed.

Example

standard input	standard output
2 2 > 3 = = 	 ? 1 2 ! 2 1 ? 3 1 ? 2 1 ! 2 3

Problem I. Ber Patio

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

Polycarp is a regular customer at the restaurant "Ber Patio". He likes having lunches there.

"Ber Patio" has special discount program for regular customers. A customer can collect bonuses and partially cover expenses in the restaurant.

Let's assume a customer currently has b bonuses and she has to pay r burles for a lunch. In this case the customer can use bonuses (1 bonus = 1 burle) to reduce the payment. She can cover at most half of the payment using bonuses. However, 1 bonus will be added to the customer's bonus balance per each 10 burles she paid.

Formally:

1. a customer can choose any number x of bonuses to use ($0 \le x \le \min(b, \lfloor \frac{r}{2} \rfloor)$),
2. the customer's bonus balance is reduced by x,
3. the customer pays $r - x$ burles,
4. the customer's bonus balance is increased by $\lfloor (r - x)/10 \rfloor$ (i.e. integer division rounded down is used).

Initially, there are b bonuses on Polycarp's account. Polycarp is going to have a lunch in "Ber Patio" for the next n days. He estimated the values $a_1, a_2, \ldots, a_n$, where a_i is the number of burles in a receipt for the i-th day. The sum over all receipts doesn't exceed 10^5 burles.

Write a program to find the minimum number of burles Polycarp has to spend and an optimal strategy to use bonuses.

Input

The first line contains two integer numbers n and b ($1 \le n \le 5000$, $0 \le b \le 10^5$) — number of days and initial number of bonuses Polycarp has.

The second line contains the integer sequence $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 1000$), where a_i is the amount of burles in the i-th day's receipt.

It is guaranteed that the sum of all receipts does not exceed 10^5 burles.

Output

On the first line, print the expected minimal number of burles to pay for all n receipts.

On the second line, print the sequence of integer numbers $b_1, b_2, \ldots, b_n$, where b_i is the number of bonuses to use on the i-th day. If there are multiple solutions, print any of them.

Examples

standard input	standard output
3 21 12 75 52	110 2 5 22
3 39 58 64 33	107 28 4 16

Problem J. Car Repair Shop

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

Polycarp starts his own business. Tomorrow will be the first working day of his car repair shop. For now the car repair shop is very small and only one car can be repaired at a given time.

Polycarp is good at marketing, so he has already collected n requests from clients. The requests are numbered from 1 to n in order they came.

The i-th request is characterized by two values: s_i — the day when a client wants to start the repair of his car, d_i — duration (in days) to repair the car. The days are enumerated from 1, the first day is tomorrow, the second day is the day after tomorrow and so on.

Polycarp is making schedule by processing requests in the order from the first to the n-th request. He schedules the i-th request as follows:

- If the car repair shop is idle for d_i days starting from s_i ($s_i, s_i + 1, \ldots, s_i + d_i - 1$), then these days are used to repair a car of the i-th client.
- Otherwise, Polycarp finds the first day x (from 1 and further) that there are d_i subsequent days when no repair is scheduled starting from x. In other words he chooses the smallest positive x that all days $x, x + 1, \ldots, x + d_i - 1$ are not scheduled for repair of any car. So, the car of the i-th client will be repaired in the range $[x, x + d_i - 1]$. It is possible that the day x when repair is scheduled to start will be less than s_i.

Given n requests, you are asked to help Polycarp schedule all of them according to the rules above.

Input

The first line contains integer n ($1 \le n \le 200$) — the number of requests from clients.

The following n lines contain requests, one request per line. The i-th request is given as the pair of integers s_i, d_i ($1 \le s_i \le 10^9$, $1 \le d_i \le 5 \cdot 10^6$), where s_i is the preferred time to start repairing the i-th car, d_i is the number of days to repair the i-th car.

The requests should be processed in the order they are given in the input.

Output

Print n lines. The i-th line should contain two integers — the start day to repair the i-th car and the finish day to repair the i-th car.

Examples

standard input	standard output
3 9 2 7 3 2 4	9 10 1 3 4 7
4 1000000000 1000000 1000000000 1000000 100000000 1000000 1000000000 1000000	1000000000 1000999999 1 1000000 100000000 100999999 1000001 2000000

Problem K. Toda 2

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

A group of n friends enjoys playing popular video game Toda 2. There is a rating system describing skill level of each player, initially the rating of the i-th friend is r_i.

The friends decided to take part in the championship as a team. But they should have equal ratings to be allowed to compose a single team consisting of all n friends. So the friends are faced with the problem: how to make all their ratings equal.

One way to change ratings is to willingly lose in some matches. Friends can form a party consisting of **two** to **five** (but not more than n) friends and play a match in the game. When the party loses, the rating of each of its members decreases by 1. A rating can't become negative, so $r_i = 0$ doesn't change after losing.

The friends can take part in multiple matches, each time making a party from any subset of friends (but remember about constraints on party size: from 2 to 5 members).

The friends want to make their ratings equal but as high as possible.

Help the friends develop a strategy of losing the matches so that all their ratings become equal and the resulting rating is maximum possible.

Input

The first line contains a single integer n ($2 \le n \le 100$) — the number of friends.

The second line contains n non-negative integers $r_1, r_2, \ldots, r_n$ ($0 \le r_i \le 100$), where r_i is the initial rating of the i-th friend.

Output

In the first line, print a single integer R — the final rating of each of the friends.

In the second line, print integer t — the number of matches the friends have to play. Each of the following t lines should contain n characters '`0`' or '`1`', where the j-th character of the i-th line is equal to:

- '`0`', if friend j should not play in match i,
- '`1`', if friend j should play in match i.

Each line should contain between two and five characters '`1`', inclusive.

The value t should not exceed 10^4, it is guaranteed that such solution exists.

Remember that you shouldn't minimize the value t, but you should maximize R. If there are multiple solutions, print any of them.

Examples

standard input	standard output
5 4 5 1 7 4	1 8 01010 00011 01010 10010 00011 11000 00011 11000
2 1 2	0 2 11 11
3 1 1 1	1 0

Problem L. Roads Orientation Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

Berland consists of n cities and m bidirectional roads connecting pairs of cities. There is no road connecting a city to itself, and between any pair of cities there is no more than one road. It is possible to reach any city from any other moving along roads.

Currently Mr. President is in the city s and his destination is the city t. He plans to move along roads from s to t ($s \neq t$).

That's why Ministry of Fools and Roads has difficult days. The minister is afraid that Mr. President could get into a traffic jam or get lost. Who knows what else can happen!

To be sure that everything goes as planned, the minister decided to temporarily make all roads one-way. So each road will be oriented in one of two possible directions. The following conditions must be satisfied:

- There should be no cycles along roads after orientation.
- The city s should be the only such city that all its roads are oriented out (i.e. there are no ingoing roads to the city s and the city s is the only such city).
- The city t should be the only such city that all its roads are oriented in (i.e. there are no outgoing roads from the city t and the city t is the only such city).

Help the minister solve his problem. Write a program to find any such orientation of all roads or report that no solution exists.

Input

Each test in this problem contains one or more test cases to solve. The first line of the input contains positive number T — the number of cases to solve.

Each case starts with a line containing four integers n, m, s and t ($2 \leq n \leq 4 \cdot 10^5$, $1 \leq m \leq 10^6$, $1 \leq s, t \leq n$, $s \neq t$) — the number of cities, the number of roads and indices of departure and destination cities. The cities are numbered from 1 to n.

The following m lines contain roads, one road per line. Each road is given as two integer numbers x_j, y_j ($1 \leq x_j, y_j \leq n$, $x_j \neq y_j$), which means that the j-th road connects cities x_j and y_j. There is at most one road between any pair of cities. It is possible to reach any city from any other moving along roads.

The sum of values n over all cases in a test doesn't exceed $4 \cdot 10^5$. The sum of values m over all cases in a test doesn't exceed 10^6.

Output

For each case print "`Yes`" if the answer exists. In the following m lines print roads in the required directions. You can print roads in arbitrary order. If there are multiple answers, print any of them.

Print the only line "`No`" if there is no answer for a case.

Example

standard input	standard output
2 4 4 1 2 1 2 2 3 3 4 4 1 3 2 1 3 3 1 2 3	Yes 1 2 3 2 4 3 1 4 No