Problem A. Union of Doubly Linked Lists

Time limit:	2 seconds
Memory limit:	256 megabytes

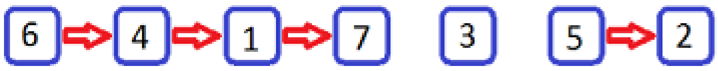
Doubly linked list is one of the fundamental data structures. A doubly linked list is a sequence of elements, each containing information about the previous and the next elements of the list. In this problem all lists have linear structure. I.e. each element except the first has exactly one previous element, each element except the last has exactly one next element. The list is not closed in a cycle.

In this problem you are given n memory cells forming one or more doubly linked lists. Each cell contains information about element from some list. Memory cells are numbered from 1 to n.

For each cell i you are given two values:

- l_i — cell containing previous element for the element in the cell i;
- r_i — cell containing next element for the element in the cell i.

If cell i contains information about the element which has no previous element then $l_i = 0$. Similarly, if cell i contains information about the element which has no next element then $r_i = 0$.



Three lists are shown on the picture.

For example, for the picture above the values of l and r are the following: $l_1 = 4$, $r_1 = 7$; $l_2 = 5$, $r_2 = 0$; $l_3 = 0$, $r_3 = 0$; $l_4 = 6$, $r_4 = 1$; $l_5 = 0$, $r_5 = 2$; $l_6 = 0$, $r_6 = 4$; $l_7 = 1$, $r_7 = 0$.

Your task is to unite all given lists in a single list, joining them to each other in any order. In particular, if the input data already contains a single list, then there is no need to perform any actions. Print the resulting list in the form of values l_i, r_i.

Any other action, other than joining the beginning of one list to the end of another, can not be performed.

Input

The first line contains a single integer n ($1 \le n \le 100$) — the number of memory cells where the doubly linked lists are located.

Each of the following n lines contains two integers l_i, r_i ($0 \le l_i, r_i \le n$) — the cells of the previous and the next element of list for cell i. Value $l_i = 0$ if element in cell i has no previous element in its list. Value $r_i = 0$ if element in cell i has no next element in its list.

It is guaranteed that the input contains the correct description of a single or more doubly linked lists. All lists have linear structure: each element of list except the first has exactly one previous element; each element of list except the last has exactly one next element. Each memory cell contains information about one element from some list, each element of each list written in one of n given cells.

Output

Print n lines, the i-th line must contain two integers l_i and r_i — the cells of the previous and the next element of list for cell i after all lists from the input are united in a single list. If there are many solutions print any of them.

Example

standard input	standard output

7
4 7
5 0
0 0
6 1
0 2
0 4
1 0
	4 7
	5 6
	0 5
	6 1
	3 2
	2 4
	1 0

Problem B. Preparing for Merge Sort

Time limit:	2 seconds
Memory limit:	256 megabytes

Ivan has an array consisting of n different integers. He decided to reorder all elements in increasing order. Ivan loves merge sort so he decided to represent his array with one or several increasing sequences which he then plans to merge into one sorted array.

Ivan represent his array with increasing sequences with help of the following algorithm.

While there is at least one unused number in array Ivan repeats the following procedure:

- iterate through array from the left to the right;
- Ivan only looks at unused numbers on current iteration;
- if current number is the first unused number on this iteration or this number is greater than previous unused number on current iteration, then Ivan marks the number as used and writes it down.

For example, if Ivan's array looks like `[1, 3, 2, 5, 4]` then he will perform two iterations. On first iteration Ivan will use and write numbers `[1, 3, 5]`, and on second one — `[2, 4]`.

Write a program which helps Ivan and finds representation of the given array with one or several increasing sequences in accordance with algorithm described above.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of elements in Ivan's array.

The second line contains a sequence consisting of distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — Ivan's array.

Output

Print representation of the given array in the form of one or more increasing sequences in accordance with the algorithm described above. Each sequence must be printed on a new line.

Examples

standard input	standard output
5 1 3 2 5 4	1 3 5 2 4
4 4 3 2 1	4 3 2 1
4 10 30 50 101	10 30 50 101

Problem C. Sum of Nestings

Time limit:	2 seconds
Memory limit:	256 megabytes

Recall that the bracket sequence is considered regular if it is possible to insert symbols '+' and '1' into it so that the result is a correct arithmetic expression. For example, a sequence "`(()())`" is regular, because we can get correct arithmetic expression insering symbols '+' and '1': "`((1+1)+(1+1))`". Also the following sequences are regular: "`()()()`", "`(())`" and "`()`". The following sequences are not regular bracket sequences: "`)(`", "`(()`" and "`())(()`".

In this problem you are given two integers n and k. Your task is to construct a regular bracket sequence consisting of round brackets with length $2 \cdot n$ with total sum of nesting of all opening brackets equals to exactly k. The nesting of a single opening bracket equals to the number of pairs of brackets in which current opening bracket is embedded.

For example, in the sequence "`()(())`" the nesting of first opening bracket equals to 0, the nesting of the second opening bracket equals to 0 and the nesting of the third opening bracket equal to 1. So the total sum of nestings equals to 1.

Input

The first line contains two integers n and k ($1 \le n \le 3 \cdot 10^5$, $0 \le k \le 10^{18}$) — the number of opening brackets and needed total nesting.

Output

Print the required regular bracket sequence consisting of round brackets.

If there is no solution print "`Impossible`" (without quotes).

Examples

standard input	standard output
`3 1`	`()(())`
`4 6`	`(((())))`
`2 5`	`Impossible`

Note

The first example is examined in the statement.

In the second example the answer is "`(((())))`". The nesting of the first opening bracket is 0, the nesting of the second is 1, the nesting of the third is 2, the nesting of fourth is 3. So the total sum of nestings equals to $0 + 1 + 2 + 3 = 6$.

In the third it is impossible to construct a regular bracket sequence, because the maximum possible total sum of nestings for two opening brackets equals to 1. This total sum of nestings is obtained for the sequence "`(())`".

Problem D. Dog Show

Time limit:	2 seconds
Memory limit:	256 megabytes

A new dog show on TV is starting next week. On the show dogs are required to demonstrate bottomless stomach, strategic thinking and self-preservation instinct. You and your dog are invited to compete with other participants and naturally you want to win!

On the show a dog needs to eat as many bowls of dog food as possible (bottomless stomach helps here). Dogs compete separately of each other and the rules are as follows:

At the start of the show the dog and the bowls are located on a line. The dog starts at position $x = 0$ and n bowls are located at positions $x = 1, x = 2, \ldots, x = n$. The bowls are numbered from 1 to n from left to right. After the show starts the dog immediately begins to run to the right to the first bowl.

The food inside bowls is not ready for eating at the start because it is too hot (dog's self-preservation instinct prevents eating). More formally, the dog can eat from the i-th bowl after t_i seconds from the start of the show or later.

It takes dog 1 second to move from the position x to the position $x + 1$. The dog is not allowed to move to the left, the dog runs only to the right with the constant speed 1 distance unit per second. When the dog reaches a bowl (say, the bowl i), the following cases are possible:

- the food had cooled down (i.e. it passed at least t_i seconds from the show start): the dog immediately eats the food and runs to the right without any stop,
- the food is hot (i.e. it passed less than t_i seconds from the show start): the dog has two options: to wait for the i-th bowl, eat the food and continue to run at the moment t_i or to skip the i-th bowl and continue to run to the right without any stop.

After T seconds from the start the show ends. If the dog reaches a bowl of food at moment T the dog can not eat it. The show stops before T seconds if the dog had run to the right of the last bowl.

You need to help your dog create a strategy with which the maximum possible number of bowls of food will be eaten in T seconds.

Input

Two integer numbers are given in the first line - n and T ($1 \le n \le 200\,000$, $1 \le T \le 2 \cdot 10^9$) — the number of bowls of food and the time when the dog is stopped.

On the next line numbers $t_1, t_2, \ldots, t_n$ ($1 \le t_i \le 10^9$) are given, where t_i is the moment of time when the i-th bowl of food is ready for eating.

Output

Output a single integer — the maximum number of bowls of food the dog will be able to eat in T seconds.

Examples

standard input	standard output
3 5 1 5 3	2
1 2 1	1
1 1 1	0

Note

In the first example the dog should skip the second bowl to eat from the two bowls (the first and the third).

Problem E. Packmen

Time limit:	1 second
Memory limit:	256 megabytes

A game field is a strip of $1 \times n$ square cells. In some cells there are Packmen, in some cells — asterisks, other cells are empty.

Packman can move to neighboring cell in 1 time unit. If there is an asterisk in the target cell then Packman eats it. Packman doesn't spend any time to eat an asterisk.

In the initial moment of time all Packmen begin to move. Each Packman can change direction of its move unlimited number of times, but it is not allowed to go beyond the boundaries of the game field. Packmen do not interfere with the movement of other packmen; in one cell there can be any number of packmen moving in any directions.

Your task is to determine minimum possible time after which Packmen can eat all the asterisks.

Input

The first line contains a single integer n ($2 \le n \le 10^5$) — the length of the game field.

The second line contains the description of the game field consisting of n symbols. If there is symbol '`.`' in position i — the cell i is empty. If there is symbol '`*`' in position i — in the cell i contains an asterisk. If there is symbol '`P`' in position i — Packman is in the cell i.

It is guaranteed that on the game field there is at least one Packman and at least one asterisk.

Output

Print minimum possible time after which Packmen can eat all asterisks.

Examples

standard input	standard output
7 `*..P*P*`	3
10 `.**PP.*P.*`	2

Note

In the first example Packman in position 4 will move to the left and will eat asterisk in position 1. He will spend 3 time units on it. During the same 3 time units Packman in position 6 will eat both of neighboring with it asterisks. For example, it can move to the left and eat asterisk in position 5 (in 1 time unit) and then move from the position 5 to the right and eat asterisk in the position 7 (in 2 time units). So in 3 time units Packmen will eat all asterisks on the game field.

In the second example Packman in the position 4 will move to the left and after 2 time units will eat asterisks in positions 3 and 2. Packmen in positions 5 and 8 will move to the right and in 2 time units will eat asterisks in positions 7 and 10, respectively. So 2 time units is enough for Packmen to eat all asterisks on the game field.

Problem F. Berland Elections

Time limit:	1 second
Memory limit:	256 megabytes

The elections to Berland parliament are happening today. Voting is in full swing!

Totally there are n candidates, they are numbered from 1 to n. Based on election results k $(1 \le k \le n)$ top candidates will take seats in the parliament.

After the end of the voting the number of votes for each candidate is calculated. In the resulting table the candidates are ordered by the number of votes. In case of tie (equal number of votes) they are ordered by the time of the last vote given. The candidate with ealier last vote stands higher in the resulting table.

So in the resulting table candidates are sorted by the number of votes (more votes stand for the higher place) and if two candidates have equal number of votes they are sorted by the time of last vote (earlier last vote stands for the higher place).

There is no way for a candidate with zero votes to take a seat in the parliament. So it is possible that less than k candidates will take a seat in the parliament.

In Berland there are m citizens who can vote. Each of them will vote for some candidate. Each citizen will give a vote to exactly one of n candidates. There is no option "against everyone" on the elections. It is not accepted to spoil bulletins or not to go to elections. So each of m citizens will vote for exactly one of n candidates.

At the moment a citizens have voted already $(1 \le a \le m)$. This is an open election, so for each citizen it is known the candidate for which the citizen has voted. Formally, the j-th citizen voted for the candidate g_j. The citizens who already voted are numbered in chronological order; i.e. the $(j+1)$-th citizen voted after the j-th.

The remaining $m - a$ citizens will vote before the end of elections, each of them will vote for one of n candidates.

Your task is to determine for each of n candidates one of the three possible outcomes:

- a candidate will be elected to the parliament regardless of votes of the remaining $m - a$ citizens;
- a candidate has chance to be elected to the parliament after all n citizens have voted;
- a candidate has no chances to be elected to the parliament regardless of votes of the remaining $m-a$ citizens.

Input

The first line contains four integers n, k, m and a $(1 \le k \le n \le 100,\ 1 \le m \le 100,\ 1 \le a \le m)$ — the number of candidates, the number of seats in the parliament, the number of Berland citizens and the number of citizens who already have voted.

The second line contains a sequence of a integers $g_1, g_2, \dots, g_a$ $(1 \le g_j \le n)$, where g_j is the candidate for which the j-th citizen has voted. Citizens who already voted are numbered in increasing order of voting times.

Output

Print the sequence consisting of n integers $r_1, r_2, \dots, r_n$ where:

- $r_i = 1$ means that the i-th candidate is guaranteed to take seat in the parliament regardless of votes of the remaining $m - a$ citizens;
- $r_i = 2$ means that the i-th candidate has a chance to take a seat in the parliament, i.e. the remaining $m - a$ citizens can vote in such a way that the candidate will take a seat in the parliament;

- $r_i = 3$ means that the i-th candidate will not take a seat in the parliament regardless of votes of the remaining $m - a$ citizens.

Examples

standard input	standard output
3 1 5 4 1 2 1 3	1 3 3
3 1 5 3 1 3 1	2 3 2
3 2 5 3 1 3 1	1 2 2

Problem G. University Classes

Time limit:	1 second
Memory limit:	256 megabytes

There are n student groups at the university. During the study day, each group can take no more than 7 classes. Seven time slots numbered from 1 to 7 are allocated for the classes.

The schedule on Monday is known for each group, i. e. time slots when group will have classes are known.

Your task is to determine the minimum number of rooms needed to hold classes for all groups on Monday. Note that one room can hold at most one group class in a single time slot.

Input

The first line contains a single integer n ($1 \le n \le 1000$) — the number of groups.

Each of the following n lines contains a sequence consisting of 7 zeroes and ones — the schedule of classes on Monday for a group. If the symbol in a position equals to 1 then the group has class in the corresponding time slot. In the other case, the group has no class in the corresponding time slot.

Output

Print minimum number of rooms needed to hold all groups classes on Monday.

Examples

standard input	standard output
2 0101010 1010101	1
3 0101011 0011001 0110111	3

Note

In the first example one room is enough. It will be occupied in each of the seven time slot by the first group or by the second group.

In the second example three rooms is enough, because in the seventh time slot all three groups have classes.

Problem H. Load Testing

Time limit:	1 second
Memory limit:	256 megabytes

Polycarp plans to conduct a load testing of its new project Fakebook. He already agreed with his friends that at certain points in time they will send requests to Fakebook. The load testing will last n minutes and in the i-th minute friends will send a_i requests.

Polycarp plans to test Fakebook under a special kind of load. In case the information about Fakebook gets into the mass media, Polycarp hopes for a monotone increase of the load, followed by a monotone decrease of the interest to the service. Polycarp wants to test this form of load.

Your task is to determine how many requests Polycarp must add so that before some moment the load on the server strictly increases and after that moment strictly decreases. Both the increasing part and the decreasing part can be empty (i. e. absent). The decrease should immediately follow the increase. In particular, the load with two equal neigbouring values is unacceptable.

For example, if the load is described with one of the arrays `[1, 2, 8, 4, 3]`, `[1, 3, 5]` or `[10]`, then such load satisfies Polycarp (in each of the cases there is an increasing part, immediately followed with a decreasing part). If the load is described with one of the arrays `[1, 2, 2, 1]`, `[2, 1, 2]` or `[10, 10]`, then such load does not satisfy Polycarp.

Help Polycarp to make the minimum number of additional requests, so that the resulting load satisfies Polycarp. He can make any number of additional requests at any minute from 1 to n.

Input

The first line contains a single integer n ($1 \le n \le 100\,000$) — the duration of the load testing.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the number of requests from friends in the i-th minute of the load testing.

Output

Print the minimum number of additional requests from Polycarp that would make the load strictly increasing in the beginning and then strictly decreasing afterwards.

Examples

standard input	standard output
5 1 4 3 2 5	6
5 1 2 2 2 1	1
7 10 20 40 50 70 90 30	0

Note

In the first example Polycarp must make two additional requests in the third minute and four additional requests in the fourth minute. So the resulting load will look like: `[1, 4, 5, 6, 5]`. In total, Polycarp will make 6 additional requests.

In the second example it is enough to make one additional request in the third minute, so the answer is 1.

In the third example the load already satisfies all conditions described in the statement, so the answer is 0.

Problem I. Noise Level

Time limit:	5 seconds
Memory limit:	256 megabytes

The Berland's capital has the form of a rectangle with sizes $n \times m$ quarters. All quarters are divided into three types:

- regular (labeled with the character '`.`') — such quarters do not produce the noise but are not obstacles to the propagation of the noise;
- sources of noise (labeled with an uppercase Latin letter from '`A`' to '`Z`') — such quarters are noise sources and are not obstacles to the propagation of the noise;
- heavily built-up (labeled with the character '`*`') — such quarters are soundproofed, the noise does not penetrate into them and they themselves are obstacles to the propagation of noise.

A quarter labeled with letter '`A`' produces q units of noise. A quarter labeled with letter '`B`' produces $2 \cdot q$ units of noise. And so on, up to a quarter labeled with letter '`Z`', which produces $26 \cdot q$ units of noise. There can be any number of quarters labeled with each letter in the city.

When propagating from the source of the noise, the noise level is halved when moving from one quarter to a quarter that shares a side with it (when an odd number is to be halved, it's rounded down). The noise spreads along the chain. For example, if some quarter is located at a distance 2 from the noise source, then the value of noise which will reach the quarter is divided by 4. So the noise level that comes from the source to the quarter is determined solely by the length of the shortest path between them. Heavily built-up quarters are obstacles, the noise does not penetrate into them.

.	.	.
A	*	.
.	B	.

$\Rightarrow$

50+25=75	25+12=37	12+25=37
100+50=150	0	6+50=56
50+100=150	25+200=225	12+100=112

The values in the cells of the table on the right show the total noise level in the respective quarters for $q = 100$, the first term in each sum is the noise from the quarter '`A`', the second — the noise from the quarter '`B`'.

The noise level in quarter is defined as the sum of the noise from all sources. To assess the quality of life of the population of the capital of Berland, it is required to find the number of quarters whose noise level exceeds the allowed level p.

Input

The first line contains four integers n, m, q and p ($1 \le n, m \le 250$, $1 \le q, p \le 10^6$) — the sizes of Berland's capital, the number of noise units that a quarter '`A`' produces, and the allowable noise level.

Each of the following n lines contains m characters — the description of the capital quarters, in the format that was described in the statement above. It is possible that in the Berland's capital there are no quarters of any type.

Output

Print the number of quarters, in which the noise level exceeds the allowed level p.

Examples

standard input	standard output
`3 3 100 140`	`3`
`...`	
`A*.`	
`.B.`	
`3 3 2 8`	`4`
`B*.`	
`BB*`	
`BBB`	
`3 4 5 4`	`7`
`..*B`	
`..**`	
`D...`	

Note

The illustration to the first example is in the main part of the statement.

Problem J. Students Initiation

Time limit:	2 seconds
Memory limit:	256 megabytes

Soon the first year students will be initiated into students at the University of Berland. The organizers of the initiation come up with a program for this holiday. In their opinion, it would be good if the first-year students presented small souvenirs to each other. When they voiced this idea to the first-year students, they found out the following:

- some pairs of the new students already know each other;
- each new student agrees to give souvenirs only to those with whom they are already familiar;
- each new student does not want to present too many souvenirs.

The organizers have written down all the pairs of first-year friends who are familiar with each other and now want to determine for each new student, whom they should give souvenirs to. In their opinion, in each pair of familiar students *exactly one* student must present a souvenir to another student.

First year students already decided to call the unluckiest the one who will have to present the greatest number of souvenirs. The organizers in return promised that the unluckiest will be unlucky to the minimum possible degree: of course, they will have to present the greatest number of souvenirs compared to the other students, but this number will be as small as possible.

Organizers are very busy, and they asked you to determine for each pair of first-year friends who and to whom should present a souvenir.

Input

The first line contains two integers n and m ($1 \leq n \leq 5000$, $0 \leq m \leq \min(5000, n \cdot (n-1)/2)$) — the number of the first year students and the number of pairs of the students that know each other. The students are numbered from 1 to n.

Each of the following m lines contains two integers x_i, y_i ($1 \leq x_i, y_i \leq n$, $x_i \neq y_i$) — the students in each pair.

It is guaranteed that each pair is present in the list exactly once. It is also guaranteed that if there is a pair (x_i, y_i) in the list, then there is no pair (y_i, x_i).

Output

Print a single integer into the first line — the smallest number of souvenirs that the unluckiest student will have to present.

Following should be m lines, each containing two integers — the students which are familiar with each other. The first number in the pair must be the student that will present the souvenir to the second student in the pair.

Pairs can be printed in any order. If there are many solutions, print any of them.

Examples

standard input	standard output

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

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

4 6
1 2
4 1
4 2
3 2
4 3
1 3
	2
	1 3
	2 1
	2 4
	3 2
	4 1
	4 3

Problem K. Travel Cards

Time limit:	4 seconds
Memory limit:	256 megabytes

In the evening Polycarp decided to analyze his today's travel expenses on public transport.

The bus system in the capital of Berland is arranged in such a way that each bus runs along the route between two stops. Each bus has no intermediate stops. So each of the buses continuously runs along the route from one stop to the other and back. There is at most one bus running between a pair of stops.

Polycarp made n trips on buses. About each trip the stop where he started the trip and the the stop where he finished are known. The trips follow in the chronological order in Polycarp's notes.

It is known that one trip on any bus costs a burles. In case when passenger makes a transshipment the cost of trip decreases to b burles ($b < a$). A passenger makes a transshipment if the stop on which he boards the bus coincides with the stop where he left the previous bus. Obviously, the first trip can not be made with transshipment.

For example, if Polycarp made three consecutive trips: "BerBank" $\rightarrow$ "University", "University" $\rightarrow$ "BerMall", "University" $\rightarrow$ "BerBank", then he payed $a + b + a = 2a + b$ burles. From the BerBank he arrived to the University, where he made transshipment to the other bus and departed to the BerMall. Then he walked to the University and returned to the BerBank by bus.

Also Polycarp can buy no more than k travel cards. Each travel card costs f burles. The travel card for a single bus route makes free of charge any trip by this route (in both directions). Once purchased, a travel card can be used any number of times in any direction.

What is the smallest amount of money Polycarp could have spent today if he can buy no more than k travel cards?

Input

The first line contains five integers n, a, b, k, f ($1 \le n \le 300$, $1 \le b < a \le 100$, $0 \le k \le 300$, $1 \le f \le 1000$) where:

- n — the number of Polycarp trips,
- a — the cost of a regualar single trip,
- b — the cost of a trip after a transshipment,
- k — the maximum number of travel cards Polycarp can buy,
- f — the cost of a single travel card.

The following n lines describe the trips in the chronological order. Each line contains exactly two different words separated by a single space — the name of the start stop and the name of the finish stop of the trip. All names consist of uppercase and lowercase English letters and have lengths between 1 to 20 letters inclusive. Uppercase and lowercase letters should be considered different.

Output

Print the smallest amount of money Polycarp could have spent today, if he can purchase no more than k travel cards.

Examples

standard input	standard output
`3 5 3 1 8`	`11`
`BerBank University`	
`University BerMall`	
`University BerBank`	
`4 2 1 300 1000`	`5`
`a A`	
`A aa`	
`aa AA`	
`AA a`	

Note

In the first example Polycarp can buy travel card for the route “BerBank $\longleftrightarrow$ University” and spend 8 burles. Note that his second trip “University” $\rightarrow$ “BerMall” was made after transshipment, so for this trip Polycarp payed 3 burles. So the minimum total sum equals to $8 + 3 = 11$ burles.

In the second example it doesn’t make sense to buy travel cards. Note that each of Polycarp trip (except the first) was made with transshipment. So the minimum total sum equals to $2 + 1 + 1 + 1 = 5$ burles.

Problem L. Berland SU Computer Network

Time limit:	2 seconds
Memory limit:	256 megabytes

In the computer network of the Berland State University there are n routers numbered from 1 to n. Some pairs of routers are connected by patch cords. Information can be transmitted over patch cords in both direction. The network is arranged in such a way that communication between any two routers (directly or through other routers) is possible. There are no cycles in the network, so there is only one path between each pair of routers over patch cords.

Unfortunately, the exact topology of the network was lost by administrators. In order to restore it, the following auxiliary information was collected.

For each patch cord p, directly connected to the router i, list of routers located behind the patch cord p relatively i is known. In other words, all routers path from which to the router i goes through p are known. So for each router i there are k_i lists, where k_i is the number of patch cords connected to i.

For example, let the network consists of three routers connected in chain $1 - 2 - 3$. Then:

- the router 1: for the single patch cord connected to the first router there is a single list containing two routers: 2 and 3;
- the router 2: for each of the patch cords connected to the second router there is a list: one list contains the router 1 and the other — the router 3;
- the router 3: for the single patch cord connected to the third router there is a single list containing two routers: 1 and 2.

Your task is to help administrators to restore the network topology, i. e. to identify all pairs of routers directly connected by a patch cord.

Input

The first line contains a single integer n ($2 \le n \le 1000$) — the number of routers in the network.

The i-th of the following n lines contains a description of the lists for the router i.

The description of each list begins with the number of routers in it. Then the symbol '`:`' follows, and after that the numbers of routers from the list are given. This numbers are separated by comma. Lists are separated by symbol '`-`'.

It is guaranteed, that for each router i the total number of routers in its lists equals to $n - 1$ and all the numbers in lists of each router are distinct. For each router i lists do not contain the number i.

Output

Print `-1` if no solution exists.

In the other case print to the first line $n - 1$ — the total number of patch cords in the network. In each of the following $n - 1$ lines print two integers — the routers which are directly connected by a patch cord. Information about each patch cord must be printed exactly once.

Patch cords and routers can be printed in arbitrary order.

Examples

standard input	standard output

3
2:3,2
1:1-1:3
2:1,2
	2
	2 1
	2 3

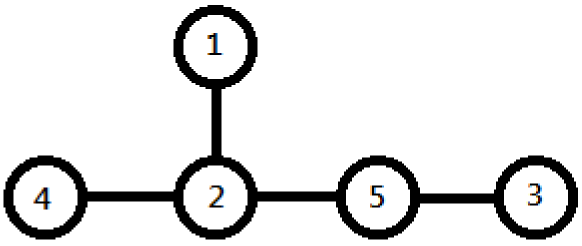
5
4:2,5,3,4
1:4-1:1-2:5,3
4:4,5,2,1
4:2,1,3,5
1:3-3:4,2,1
	4
	2 1
	2 4
	5 2
	3 5

3 1:2-1:3 1:1-1:3 1:1-1:2	-1

Note

The first example is analyzed in the statement.

The answer to the second example is shown on the picture.



The first router has one list, which contains all other routers. The second router has three lists: the first — the single router 4, the second — the single router 1, the third — two routers 3 and 5. The third router has one list, which contains all other routers. The fourth router also has one list, which contains all other routers. The fifth router has two lists: the first — the single router 3, the second — three routers 1, 2 and 4.

Problem M. Weather Tomorrow

Time limit:	1 second
Memory limit:	256 megabytes

Vasya came up with his own weather forecasting method. He knows the information about the average air temperature for each of the last n days. Assume that the average air temperature for each day is integral.

Vasya believes that if the average temperatures over the last n days form an arithmetic progression, where the first term equals to the average temperature on the first day, the second term equals to the average temperature on the second day and so on, then the average temperature of the next $(n+1)$-th day will be equal to the next term of the arithmetic progression. Otherwise, according to Vasya's method, the temperature of the $(n+1)$-th day will be equal to the temperature of the n-th day.

Your task is to help Vasya predict the average temperature for tomorrow, i. e. for the $(n+1)$-th day.

Input

The first line contains a single integer n ($2 \le n \le 100$) — the number of days for which the average air temperature is known.

The second line contains a sequence of integers $t_1, t_2, \ldots, t_n$ ($-1000 \le t_i \le 1000$) — where t_i is the average temperature in the i-th day.

Output

Print the average air temperature in the $(n+1)$-th day, which Vasya predicts according to his method. Note that the absolute value of the predicted temperature can exceed 1000.

Examples

standard input	standard output
5 10 5 0 -5 -10	-15
4 1 1 1 1	1
3 5 1 -5	-5
2 900 1000	1100

Note

In the first example the sequence of the average temperatures is an arithmetic progression where the first term is 10 and each following terms decreases by 5. So the predicted average temperature for the sixth day is $-10 - 5 = -15$.

In the second example the sequence of the average temperatures is an arithmetic progression where the first term is 1 and each following terms equals to the previous one. So the predicted average temperature in the fifth day is 1.

In the third example the average temperatures do not form an arithmetic progression, so the average temperature of the fourth day equals to the temperature of the third day and equals to -5.

In the fourth example the sequence of the average temperatures is an arithmetic progression where the first term is 900 and each the following terms increase by 100. So predicted average temperature in the third day is $1000 + 100 = 1100$.