Problem A. Cafe

Time limit:	2 seconds
Memory limit:	256 megabytes

After competing in a programming contest n groups of friends have decided to take a rest from problem solving and have come to a cafe. The groups are numbered by integers from 1 to n, and the i-th group contains a_i friends. The tables in the cafe are absolutely identical, for r persons each. Now the friends have faced a tricky problem. They want to sit at the tables in such a way that nobody sits alone separately from his or her group. In other words, for each person there should be another person from his/her group sitting at his/her table. There are no other restrictions. People from different groups can sit at the same table. Friends from the same group can sit at different tables. Places at the tables can be left vacant. Your task is to seat all the friends in the described way using the minimum possible number of tables.

Input

The first line of the input contains two integers n and r ($1 \le n \le 2000$; $3 \le r \le 2000$), where n is the number of groups, and r is the number of places at each table. The second line contains n space-separated integers a_i ($2 \le a_i \le 2000$) — the number of people in the i-th group.

Output

Print the minimum possible number of tables to seat all the friends.

Examples

input.txt	output.txt
3 4 5 6 7	5
4 4 3 3 3 3	4

Note

A possible arrangement for the first sample test is:

- table 1: 3 people from the 1st group, 1 empty place;
- table 2: 2 people from the 1st group, 2 people from the 2nd group;
- table 3: 4 people from the 2nd group;
- table 4: 4 people from the 3rd group;
- table 5: 3 people from the 3rd group, 1 empty place.

Problem B. Chess Championship

Time limit:	6 seconds
Memory limit:	256 megabytes

Chess Championship is set up in the New Basyuki City. Two national teams, of Berland and Byteland, are going to have a match there. Each team is represented by n players. The championship consists of n games — in each game a pair of players from different teams meets. A game victory brings 1 point to the team and a game defeat doesn't add or subtract points.

The championship starts with the sortition — the process that determines opponent pairs. Each player from the first team plays with exactly one player from the second team (and vice versa).

A recent research conducted by Berland scientists showed that every player of either team is characterized by a single number — the level of his chess mastership. No two people among the $2n$ players play at the same level. Funny as it is, the game winner always is the player with the higher level.

The contest organizers received information that a high-ranking Berland official Mr. B. bet 100500 burles on the victory of his team with a k points gap. Immediately an unofficial "recommendation" came from very important people to "organize" the sortition so that the Berland team gets exactly k points more than the Byteland team.

Write a program that finds the number of distinct sortition results after which Berland gets exactly k points more than Byteland. Two sortitions are considered distinct if there is such player, that gets different opponents by the sortitions' results.

Input

The first line contains a pair of integers n and k ($1 \le n \le 500$; $1 \le k \le n$) — the number of participants in each team and the required gap in points that Berland must win. The second and the third input lines contain n integers each: the i-th number of the second line characterizes the i-th Berland player's chess mastership level, and the j-th number of the third line characterizes the j-th Byteland player's chess mastership level. It is guaranteed that all numbers that characterize mastership levels are distinct integers from 0 to 10^9.

Output

Print a single integer — the number of ways to set up the sortition so that the Berland team wins k points more than the Byteland team in the championship. The answer can be rather large, so print it modulo 1000000009 ($10^9 + 9$).

Examples

input.txt	output.txt

4 2
5 35 15 45
40 20 10 30
	4

2 2
3 4
1 2
	2

Note

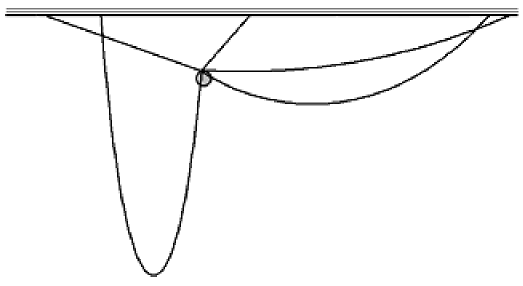
In the first example the acceptable sortition results are: (5-40, 35-20, 15-10, 45-30), (5-40, 45-20, 15-10, 35-30), (45-40, 5-20, 15-10, 35-30) and (45-40, 35-20, 15-10, 5-30).

Problem C. Cut the rope, another rope and so on!

Time limit:	2 seconds
Memory limit:	256 megabytes

A game called "Cut the rope, another rope and so on!" is very popular in Berland. People all over the country just cut ropes on their bPads and bPhones over and over again.

The game is about sequentially cutting ropes with a small ball hanging on their ends. At the beginning of the game all ropes have one end attached to the top of the screen, that is, we can assume that the ends are located on the horizontal axis Ox. The other ends of the ropes are connected together and a small ball is attached to them. The ropes never shrink or stretch. At the beginning of the game the system is in a static state — some ropes are tight, some might be loose. In this game, the ball should be considered as a material point, that is, we assume that it is infinitely small.



Let's suppose that at the beginning the ball is suspended on n ropes. A player cuts off $n-1$ uncut ropes, one after another. The ball may move in some way after some ropes are cut. The next cutting is made as soon as the ball becomes static after the previous cutting (the first cutting is made immediately). If the present cutting does not change the ball's position, then the next cutting is done immediately.

The physics in this game is a bit peculiar: educated people say that the motion of the ball is going in a perfectly viscous medium. This means that at any time the ball is moving so that it reduces the potential energy in the fastest way. For example, if the ball can fall straight down, it moves exactly this way. Note that the ball never swings, that is, its movement always moves it below its current position. You should assume that ropes have negligible masses compared with the mass of the ball.

Assuming that the ball always moves at the constant speed of 1 unit of length per unit of time, find the position of the ball at the given moments of time $t_1, t_2, \ldots, t_m$.

Input

The first input line contains two integers n and m ($2 \le n \le 20$; $1 \le m \le 1000$) — the number of ropes and the number of ball position queries, correspondingly.

The second line contains n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 200$) — the coordinates of the ropes' attachment points on the horizontal axis.

The third line contains n integers $l_1, l_2, \ldots, l_n$ ($1 \le l_i \le 200$), l_i is the length of the i-th rope. It is guaranteed that values $x_1, x_2, \ldots, x_n$ and $l_1, l_2, \ldots, l_n$ are such that the initial configuration really exists, that is, the lower ends of all ropes can be simultaneously tied to a ball. It is guaranteed that no two ropes have equal lengths and attached to the same point at the same time.

The fourth line describes the sequence of cuts, it contains $n-1$ distinct integers $p_1, p_2, \ldots, p_{n-1}$ ($1 \le p_j \le n$), where p_j is the index of the rope that is the j-th one to be cut.

The last input line contains m real numbers $t_1, t_2, \ldots, t_m$ ($0 \le t_k \le 1000$) — the moments of time for determining the ball's position. All t_i are given with no more than three digits after the decimal point. The first rope is cut at time 0, each of the other ropes is cut at the moment the ball becomes stable after the cutting of the previous rope. If the current cutting doesn't change the ball's position, then the next rope is cut immediately. All values $t_1, t_2, \ldots, t_m$ are distinct and are given in the increasing order.

Output

Print m lines, the k-th line must contain the coordinates of the ball at time t_k. Consider all ropes attached to the Ox axis, and the Oy axis is directed upwards. Print the numbers with at least 5 digits after the decimal point. After the ball hangs on the last rope, it doesn't move anymore.

Examples

input.txt	output.txt

3 4
1 1 3
5 10 20
1 2
2.5 10 16 20
	1.0000000000 -7.5000000000
	1.0000000000 -15.0000000000
	2.0972097000 -19.9796138520
	3.0000000000 -20.0000000000

3 7
2 22 19
12 12 10
2 1
0 0.18 0.19 4.39 6 7 9
	12.0000000000 -6.6332495807
	11.8993800085 -6.7824977292
	11.8937244905 -6.7907448565
	15.0867736994 -9.2025355158
	16.6125973239 -9.7108345914
	17.5939901889 -9.9006634329
	19.0000000000 -10.0000000000

Problem D. Divide The Kingdom

Time limit:	2 seconds
Memory limit:	256 megabytes

Once upon a time, long, long ago there lived a King and a Queen who ruled over a distant kingdom called Berland.

Their kingdom contained n cities connected by $n - 1$ bi-directional roads. The roads were constructed in such a way that Berland citizens were able to reach any city from any other city by walking along these roads.

One day the King and the Queen decided that they want to break up the relationship. Yes, it's not common when it comes to a royal marriage, but it's something they decided to do and we have to respect their decision. Of course, now they want to divide the kingdom and start living separately from each other. Each of them wants to take a part of the existing kingdom. Moreover, they want their own parts to be very special, so their requirements are quite complicated.

So, the King wants to pick a nonempty subset of cities from the existing kingdom. He will be satisfied only if all following conditions are met:

- his subset of cities is connected (i.e. for every pair of his cities A and B, there is always a path from A to B that passes only through the King's cities),
- he does not share any cities with the Queen,
- none of his cities are directly connected by a road to any of the Queen's cities,
- if you consider distances between all pairs of his cities, the length of the longest path must be equal to D_1,
- if you consider all pairs of his cities which are located at distance D_1 from each other and then calculate the number of different cities within all these pairs, this number must not exceed C_1 (formally, if his subset contains the only city then the number of such pairs equals 0).

The Queen wants to pick a nonempty subset of cities as well. Her requirements are essentially the same as the King's ones, with the exception of the numbers she has in mind — D_2 and C_2 respectively. Obviously, she is not allowed to take cities which have been already taken by the King, her subset of cities should be connected and none of her cities should be adjacent to the King's cities.

Now, what about the remaining cities, the ones that will not belong to either the King or the Queen after the kingdom is divided? The answer is simple — they have to be destroyed along with all roads coming into them, and all people should be evacuated from these cities. Destroying the i-th city will cost p_i burles.

Can you help the King and the Queen with the separation? You task is to figure out whether it's possible to perform separation of the kingdom according to the rules described above. If the separation is possible, you have to find a way with the minimum possible cost of cities that will be destroyed.

Input

The first line of the input contains a positive integer n ($3 \leq n \leq 200$) — the number of cities in the kingdom. The second line contains integers D_1, C_1, D_2, C_2 ($0 \leq D_1, D_2 \leq n - 1$, $1 \leq C_1, C_2 \leq n$). The third line contains n integer costs p_i ($1 \leq p_i \leq 1000$). Each of the next $n - 1$ lines contains two integer numbers a_j and b_j ($1 \leq a_j, b_j \leq n$), meaning that the j-th road connects cities a_j and b_j.

Output

If there is no solution, write "`-1`" (without the quotes) in a single output line.

Otherwise, output the total cost of destroyed cities in the first line, and print the numbers of destroyed cities in the increasing order in the second line. If there are multiple solutions, you may print any of them.

Examples

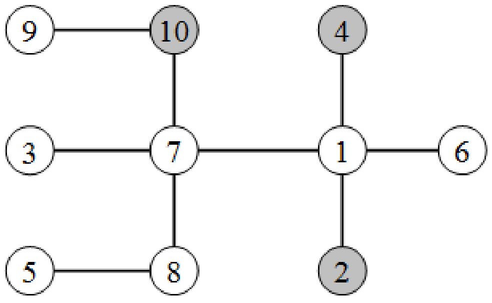
input.txt	output.txt

10
4 2 0 1
5 2 5 2 5 5 5 5 5 2
1 4
6 1
1 2
7 1
3 7
10 7
9 10
7 8
8 5
	6
	2 4 10

4
1 2 1 2
9 9 9 9
1 2
2 3
3 4
	-1

Note

In the first test case, the optimal solution is as follows:



- In the first place, city 10 is destroyed and the kingdom falls apart into two parts. The smallest part contains an isolated city 9, and it already satisfies the Queen's requirements.
- The second remaining part is a little bit too large for the King. The maximum distance between pairs of cities $(2, 5)$, $(4, 5)$, $(6, 5)$ is 4, exactly as the King wants. But the number of these cities $[2, 4, 5, 6]$ is 4, while the King's desire is to have not more than 2. So, it's additionally required to destroy cities 2 and 4.
- Overall, destroying cities $[2, 4, 10]$ costs 6 burles. It's an optimal solution from the cost perspective.

In the second test case there is no solution. Obviously, at least one city should be deleted from the kingdom, while $D_1 = 1$ requires two adjacent cities, and $D_2 = 1$ requires another two adjacent cities. So, we could possibly achieve the required outcome with 5 cities in a line, but not with 4.

Problem E. Dragons and Princesses

Time limit:	2 seconds
Memory limit:	256 megabytes

Once upon a time there lived the Knight. Being very courageous he decided to make a long journey full of fights and adventures. The map of this journey can be represented as a row with n cells numbered from 1 to n from left to right. Initially the Knight is located at the leftmost cell (cell number 1). He should pass all the cells one by one and finish his way at the rightmost cell (cell number n). He is not allowed to move back or skip some cells, he will visit all the cells from the first to the last.

Each cell except the first one contains either a dragon or a princess. Each dragon has a chest with gold coins. The dragon at the cell i keeps g_i coins. Every time the Knight steps to a cell with a dragon he has a choice — to kill the dragon or just to pass through. The Knight is very strong and dexterous, so it is not a problem for him to kill any dragon on his way. If a dragon is killed the Knight gets all the gold dragon possessed.

When the Knight steps to the cell with a princess, she wonders how many dragons he has killed. If that number is greater or equal to her beauty b_i, the princess considers the Knight brave enough and instantly asks him to marry her. Being a true gentleman, the Knight cannot refuse and his adventure immediately ends.

The Knight loves the princess who lives in the cell number n and wants to marry her. Also during the journey he wants to collect as much gold as possible. Please help him to accomplish this task.

Input

The first line of the input contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of cells. The next $n - 1$ lines describe cells from 2 to n.

If the cell number i contains a dragon, the i-th line of the input contains letter "`d`" followed by a single integer g_i ($1 \le g_i \le 10^4$) — the number of coins the dragon keeps. The letter and the integer are separated by a single space.

If the cell number i contains a princess, the i-th line of the input contains letter "`p`" followed by a single integer b_i ($1 \le b_i \le 2 \cdot 10^5$) — the beauty of the princess. The letter and the integer are separated by a single space. It is guaranteed that the last cell contains a princess.

Output

On the first line of the output print a single integer — the maximum number of gold coins the Knight can collect. On the second line print a single integer k — the number of dragons to kill. The third line should contain k integers — the numbers of the cells where the Knight should kill a dragon. The cell numbers should be printed in the increasing order.

If there are several optimal solutions, output any of them. If the Knight can't marry his beloved princess, just print `-1` in the first line of the output.

Examples

input.txt	output.txt

6
d 10
d 12
p 2
d 1
p 2
	13
	2
	3 5

6 d 10 d 12 p 2 d 1 p 3	-1

Problem F. Dumbbells

Time limit: 2 seconds
Memory limit: 256 megabytes

A sports shop has n dumbbells in store. Each of them is characterised by its mass m_i and cost c_i. Recently the shop manager faced the following non-trivial problem. He has to find the maximum number of sports sets that satisfy the following requirements:

- each set must contain exactly k dumbbells;
- each set must have dumbbells of k distinct masses;
- for each pair of sets the masses of dumbbells must coincide *completely*, that is, the masses in the sets must be equal as *mathematical sets*.

The manager's task is to make the maximum number of such sets. If there are several ways to make the maximum possible number of sets, he should choose the one that has the maximum total cost of dumbbells that are contained in the chosen sets. Note that the primary goal is to maximize the number of sets and maximization of the total cost is the secondary goal.

Input

The first line of the input contains integers n and k ($1 \le n, k \le 4000$). Next n lines contain descriptions of the dumbbells, one per line. Each description consists of a pair of integers m_i, c_i ($1 \le m_i, c_i \le 4000$), m_i is the mass of the i-th dumbbell, and c_i is its cost.

Output

In the only output line print two integers — t and s, where t is the maximum number of sets, and s is the maximum total cost of dumbbells in t choosen sets. If the manager can't make at least one set, print a pair of zeroes.

Examples

input.txt	output.txt
`7 2`	`2 22`
`16 1`	
`4 6`	
`16 7`	
`7 100`	
`32 9`	
`4 6`	
`32 1`	
`4 2`	`1 10`
`1 2`	
`2 1`	
`4 3`	
`1 7`	

Note

In the first sample the manager should make two sets. One of the possible solutions is as follows: the first set contains the second and the seventh dumbbells, the second set contains the fifth and the sixth dumbbells.

In the second sample the manager can make only one set. It consists of the third and the fourth dumbbells.

Problem G. Database optimization

Time limit:	6 seconds
Memory limit:	256 megabytes

Alex worked at a big IT-company called Macrohard. Once, when he faced a large amount of data, he decided to leave this company and develop his own database which would be much better than all existing ones. When he was done with this he realized that the performance of some database queries might be improved. Alex uses AQL (Alex Query Language) which accidentally turned out to be exactly the same as the popular SQL. One of the most important problems he faced was the following.

Consider n objects. The i-th object has k_i ($1 \le k_i \le 4$) properties in the form of `key=value`. Any object can't have two or more properties with the same key. Alex needs to improve the performance of the following query:

$$\texttt{SELECT COUNT(*) FROM Objects WHERE } key_1\texttt{=}value_1 \texttt{ AND ... AND } key_l\texttt{=}value_l$$
$$(1 \le l \le 4, \text{ all keys are distinct})$$

This means that Alex's database has to find the number of objects which have properties key_1, key_2, …, key_l with the values $value_1$, $value_2$, …, $value_l$ respectively. Even if an object has extra properties it should be counted.

Your task is to help Alex to write an efficient program to process such queries.

Input

The first line of the input contains a single integer n ($1 \le n \le 5 \cdot 10^4$) — the number of objects. Following n lines describe objects by their properties. Each line contains the integer k_i ($1 \le k_i \le 4$) followed by k_i tokens in the form of $key_{i,j}$=$value_{i,j}$ separated by a single space. Both $key_{i,j}$ and $value_{i,j}$ consist of digits and lowercase Latin letters. The $key_{i,j}$ are distinct for each object. It is possible that different objects have exactly the same set of properties.

The next line of the input contains a single integer m ($1 \le m \le 10^5$) — the number of queries. The following m lines describe the queries. Each line contains a single integer l_i (the number of properties that describe the i-th query) followed by l_i ($1 \le l_i \le 4$) tokens $key_{i,j}$=$value_{i,j}$ separated by a single space, where $key_{i,j}$ and $value_{i,j}$ consist of digits and lowercase Latin letters. The $key_{i,j}$ are distinct for each query.

Lengths of $key_{i,j}$ and $value_{i,j}$ both for objects and queries are between 1 and 5 characters inclusive.

Output

Print m lines, the i-th line should contain the result of the i-th query.

Examples

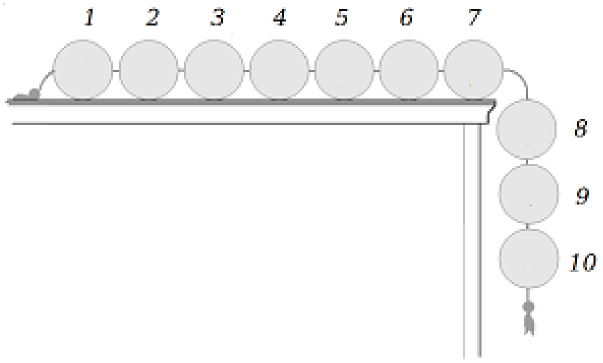
input.txt	output.txt
4 3 width=5 ht=3 len=10 2 name=circ rad=5 2 name=circ rad=5 3 name=sqr width=5 ht=3 4 2 ht=3 width=5 1 name=circ 1 name=sqr 2 width=5 ht=03	2 2 1 0

Problem H. Sultan's Pearls

Time limit:	2 seconds
Memory limit:	256 megabytes

Sultan Suleiman was so rich that legends spread far and wide about his treasures. This problem is going to be about one of those legends.

One of the sultan's favorite treasures was a string of finest pearls that he kept on the bedside table. He never touched the string as it had too many pearls on it to wear. The sultan's cunning servant decided to take advantage of this fact and "borrow" a few pearls. The string consisted of n pearls, m of them hung down from the bedside table. In this problem we will consider the pearls indexed by integers from 1 to n, starting from the end that lies on the table, that is, pearls $1, 2, \ldots, n - m$ were located on the table and pearls $n - m + 1, n - m + 2, \ldots, n$ hung down from it.



Sample for $n = 10$ and $m = 3$.

The servant decided to take exactly one pearl from one end of the string every day. But he had to be perfectly careful as every evening the sultan enjoyed looking at the string and counting the number of the hanging pearls. That's why after the servant took a pearl from the hanging end, he had to pull the string one pearl lower so that the number of the hanging pearls equalled m again. Certainly, if the servant took a pearl from the lying end, he had to leave the hanging part as it was.

Each pearl has some mass, and the string may fall down if the hanging part is too heavy. Of course, the servant must avoid that. The string must remain motionless after every action of the servant.

More formally, assume that the i-th pearl in the string has mass of w_i. Also let's say that the total mass of the hanging m pearls equals W_h, and the total mass of the pearls on the table equals W_t. Then the hanging part pulls the whole string down, if $W_h > k \cdot W_t$, where k is the coefficient of friction of the pearls against the table. The coefficient k is the same for all pearls.

The pearls on the string had not only different masses but also different prices: the i-th pearl costs c_i dinars. The servant's aim was to steal the pearls for the maximum sum and avoid the sultan's suspicions. His plan didn't come out very well: he made a mistake somewhere in his calculations, his theft was discovered and he was executed.

Nobody is going to execute you, of course, so we suggest you to solve the problem that proved to be too hard for the sultan's servant.

Input

The first line contains three integers n, m and k ($2 \le n \le 2 \cdot 10^5, 1 \le m < n, 1 \le k \le 10$). Each of the following n lines contains two integers w_i and c_i — the mass and the price of the i-th pearl ($1 \le w_i, c_i \le 1000$). It is guaranteed that initially the string is motionless, that is, the hanging part doesn't pull the whole string down.

Output

In the first line print two space-separated integers p and s — the number of pearls you can take to get the maximum sum of money, and the sum you can get. In the second line print the string consisting of p characters 'H' or 'T'. If the pearl that is the i-th to take should be taken from the hanging end, then the i-th character of the string must be 'H', otherwise — 'T'. If there are multiple optimal solutions, print any of them.

If the servant can't take any pearl, just print one line containing two zeroes. You may leave the second line empty or do not print it at all.

Examples

input.txt	output.txt
5 2 1	2 5
5 3	TT
4 2	
6 4	
3 2	
2 2	
20 7 2	11 60
3 4	HTHTHTHHHHH
8 4	
8 5	
6 14	
5 10	
3 18	
2 5	
2 4	
1 6	
3 11	
4 3	
3 5	
2 8	
4 6	
9 14	
7 2	
7 6	
6 4	
8 2	
10 5	

Note

There is the explanation to the second sample.

Initially the mass of pearls on the table was $W_t = 50$, and the mass of the hanging pearls was $W_h = 51$. However, as the coefficient of friction equals 2, the string is motionless ($50 \cdot 2 = 100 > 51$).

On the first step we take a pearl from the hanging part of the string (H), then we need to pull the string one pearl lower so that the hanging part contained 7 strings again. After that $W_t = 48$, and $W_h = 43$ (the pearl number 20 with value 5 will be stolen and the pearl number 13 will be the topmost pearl in the hanging part of the string).

On the second step we take a pearl from the end of the string that lies on the table (T). $W_h = 43$ still, $W_t = 45$, $\quad (45 \cdot 2 > 43)$, the total price of the stolen treasure is $S = 9$.

The table describes the values of W_t, W_h and S after each step.

Step	End	W_t	W_h	S
1	H	48	43	5
2	T	45	43	9
3	H	42	38	11
4	T	34	38	15
5	H	30	36	19
6	T	22	36	24
7	H	19	32	30
8	H	18	26	32
9	H	16	19	46
10	H	14	17	52
11	H	11	18	60

Note that after the 11-th step it is impossible to take any more pearls without disrupting the balance.

Problem I. Gena vs Petya

Time limit:	2 seconds
Memory limit:	256 megabytes

Gena and Petya love playing the following game with each other. There are n piles of stones, the i-th pile contains a_i stones. The players move in turns, Gena moves first. A player moves by choosing any non-empty pile and taking an arbitrary positive number of stones from it. If the move is impossible (that is, all piles are empty), then the game finishes and the current player is considered a loser.

Gena and Petya are the world famous experts in unusual games. We will assume that they play optimally.

Recently Petya started to notice that Gena wins too often. Petya decided that the problem is the unjust rules as Gena always gets to move first! To even their chances, Petya decided to cheat and take and hide some stones before the game begins. Since Petya does not want Gena to suspect anything, he will take the same number of stones x from each pile. This number x can be an arbitrary non-negative integer, strictly less that the minimum of a_i values.

Your task is to find the number of distinct numbers x such that Petya will win the game.

Input

The first line contains the number of piles n ($1 \le n \le 2 \cdot 10^5$). The second line contains n space-separated integers a_i ($1 \le a_i \le 10^{18}$) — the piles' sizes.

Output

Print the number of ways to choose x so that Petya will win the resulting game considering that both players play optimally.

Examples

input.txt	output.txt
2 3 3	3
3 3 4 5	1
4 2 7 4 1	1
4 4 6 8 10	2

Note

Consider the first example. Petya can choose any x between 0 and 2. After it Gena starts the game with two piles of equal sizes and looses the game. In the second example there is a single possible value of x, equal to 2. In the third example the sought x is also only one — it's $x = 0$. In the fourth example there are two possible values of x — they are 0 and 3.

Problem J. Ternary Password

Time limit:	2 seconds
Memory limit:	256 megabytes

In the ternary world all passwords are ternary, that is, they consist only of digits "`0`", "`1`" and "`2`". Terentius is trying to register on a famous internet service site Toogle, but the problem is, according to the security policy of this service the password must contain exactly a characters "`0`" and exactly b characters "`1`". All other characters of the password must be "`2`".

Terentius was fond of his password, he spent much time trying to remember it and now he can type it even with his eyes closed. That's the reason Terentius wants to replace the minimum number of characters in his password so that it meets the strict requirements of the Toogle policy. Terentius wants only to replace (substitute) some characters in password, he doesn't intend to perform other operations with the password.

Help Terentius find the minimum number of replacements and print the corresponding possible variant of the password to Toogle.

Input

The first line of the input contains three integers n, a and b ($1 \le n \le 200; 0 \le a, b \le 200$) — the length of Terentius's password, the expected number of characters "`0`" and the expected number of characters "`1`" in the password to Toogle. The second line contains Terentius's password. All characters in this line are digits "`0`", "`1`" or "`2`".

Output

In the first output line print t — the minimum number of replacements. In the second line print the possible variant of the Toogle password — the password that satisfies the Toogle safety policy, that differs from Terentius's password in exactly t positions. If there are several passwords, print any of them. Obviously, the length of the printed password must equal n.

It is possible that Terentius's password already meets the Toogle policy. In this case the first line must contain "`0`", and the second line must contain Terentius's password.

If the solution doesn't exist, that is, if it is impossible to get a password to Toogle if Terentius uses replacements only, then print "`-1`" in the first line of the output. In this case print empty second line or don't print the second line at all.

Examples

input.txt	output.txt

`6 1 3`
`012022`
	`2`
	`111022`

`5 5 0`
`02211`
	`4`
	`00000`

Problem K. Tree Queries Online

Time limit:	6 seconds
Memory limit:	256 megabytes

This problem is a little unusual — here you must implement an online algorithm for processing special queries. This means that your program can start reading a request only after it prints the response to the previous one. Note that this problem has standard input/output (i.e., from the keyboard and to the screen). After writing response to a request, be sure to use the stream flushing function. Otherwise you risk leaving part of your output in i/o buffer. For example, you must use the `fflush(stdout)` function in C++, call `System.out.flush()` in Java, and `flush(output)` in Pascal.

You are given a weighted undirected tree containing n vertices. The vertices are indexed by integers from 1 to n. A tree is a connected graph without cycles. Your task is to process $n-1$ requests for it. Each request means removing one of its edges. Of course, after the first request the tree is no longer a tree, but it becomes a forest (a set of trees).

After processing each request your program should output the weight of the removed edge w_i. In addition, the requests change the weights of other edges. When you remove an edge, the connected component that contains it splits into two. In the smaller (by the number of vertices) of the two resulting components the weights of the edges are multiplied by the weight w_i of the removed edge, and in the larger component the weight of the removed edge w_i is added to the weight of each edge. If both components have equal number of vertices, you should assume that the smaller one is the one that contains the vertex with the smallest index among the vertices of these two components.

All arithmetic operations must be performed modulo 99990001. That is, you only operate with numbers from 0 до 99990000, and in any arithmetic operation the result is the remainder of dividing the operation outcome by 99990001.

Input

The input to this problem must be read from the standard input stream. The first input line contains an integer n ($2 \le n \le 2 \cdot 10^5$) — the number of vertexes in the tree. The next $n-1$ lines contain descriptions of the edges of the tree, one description per line. Each description consists of three integers x_i, y_i, w_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$; $0 \le w_i \le 99990000$) — the indexes of vertexes connected by the i-th edge and the initial weight of the i-th edge. The last input line contains sequence $p_1, p_2, \ldots, p_{n-1}$ — the permutation of integers from 1 to $n-1$, where p_i is the number of the removed edge. The edges are numbered from 1 to $n-1$ in the order they follow in the input.

Remember that you can only read the next element of permutation $p_1, p_2, \ldots, p_{n-1}$ after the response to the current query is printed. During the testing the system won't let you read the data unless you follow this rule.

Output

The output must be written to the standard output. Print $n-1$ lines, the j-th line must contain the weight of the removed edge during the j-th query. Flush the output after each printed line, i.e. each time after your program writes end-of-line.

Examples

standard input	standard output
4	3
1 2 3	7
2 3 4	15
3 4 5	
1 2 3	
4	4
1 2 3	12
2 3 4	9
3 4 5	
2 1 3	

Problem L. Preparing Problem

Time limit:	2 seconds
Memory limit:	256 megabytes

It is not easy to prepare a problem for a programming contest. Petya and Vasya decided that problem "A+B" needs at least n distinct solutions to be written. It doesn't matter how many solutions each of them will write, they need to write at least n solutions in total. We know that Petya needs t_1 units of time to write a solution, and Vasya needs t_2 units of time. They start to work simultaneously at time 0. Thus, for example, Petya finishes writing his first solution at time t_1, his second solution at $2 \cdot t_1$ and so on.

Petya and Vasya are working by the same algorithm. Each time Petya (Vasya) finishes writing a solution, he checks on how many solutions have already been written up to the current time moment t. Ready solutions are the solutions that have been fully written by this time. The solutions that were fully finished exactly at time t are also considered ready. If the number of such solutions is strictly less than n, then Petya (Vasya) starts writing the next solution. If a member of the jury began working on a problem, he doesn't stop working under any circumstances, and he will surely finish it.

Petya and Vasya realize that if they act on this algorithm, they will not necessarily write exactly n solutions in total. Maybe they'll write more solutions.

Considering that Petya and Vasya work non-stop, find, how many solutions they wrote in total and the moment when the latest solution was finished. The latest solution is one which was finished last.

Input

The only input line contains three integers n, t_1 and t_2 ($1 \le n, t_1, t_2 \le 5000$).

Output

Print two integers — m and f, where m is the number of written solutions, and f is the moment when the last solution was finished.

Examples

input.txt	output.txt
5 2 3	5 6
5 2 4	6 8
3 30 50	4 100

Note

In the first sample Petya finished his solutions at time 2, 4 and 6, and Vasya — at time 3 and 6. They finished writing their last solutions simultaneously, at time 6, and at this exact moment they already had the total of 5 written solutions and stopped working.