Task: BAN
Bank Notes

Brazilian ICPC Summer School 2018, Day 1. Available memory: 64 MB. *29.01.2018*

The Byteotian Bit Bank (BBB) has the largest network of cash dispensers in the whole Byteotia. The BBB have decided to improve their dispensers and have asked you for help. The legal tender in Byteotia are bank notes of denominations $b_1, b_2, \ldots, b_n$. The BBB have concluded that the cash dispensers are to pay every sum in the smallest possible total number of notes.

Write a programme that reads the description of the dispenser's notes stock and the sum to be paid off, and determines the minimal total number of bank notes sufficient to pay the desired sum off, and finds some way of paying it off as well (using the determined minimal number of notes, of course).

Input

In the first line of the standard input the number of denominations n is written ($1 \leq n \leq 200$). The second line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_1 < b_2 < \ldots < b_n \leq 20\,000$) separated by single spaces. The third line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq 20\,000$) also separated by single spaces; c_i is the number of banknotes of denomination b_i left in the cash dispenser. In the last, fourth line of input there is one integer k – the sum to be paid off ($1 \leq k \leq 20\,000$). For the test data, you are free to assume that the sum k can be paid off in the available banknotes.

Output

The first line of the standard output should contain one integer denoting the minimal total number of bank notes sufficient to pay off the sum of k. The second line should contain n integers, separated by single spaces, denoting the numbers of notes of subsequent denominations used to pay off the sum k. If there are many solutions your programme should write any of them.

Example

For the input data:

```
3
2 3 5
2 2 1
10
```

the correct result is:

```
3
1 1 1
```

Task: NAD
Transmitters

Brazilian ICPC Summer School 2018, Day 1. Available memory: 256 MB. *29.01.2018*

Byteasar has become the new head of a historic salt mine near Bytown. To increase its popularity with the tourists, he has decided to have wireless Internet set up in the mine's corridors.

The mine consists of n chambers connected by $n-1$ corridors. Each chamber can be reached from every other using the corridors. Byteasar wants to have the wi-fi transmitters installed in the chambers so that Internet is available in every corridor of the mine. The signal in the corridor linking chambers a and b will be sufficiently strong if at least one of the following conditions holds:

- there is a transmitter in chamber a or chamber b, or
- inside the chambers that can be reached from chamber a or b using at most one corridor, there is a total of at least two transmitters.

Byteasar wonders how many wi-fi transmitters are required to provide sufficiently strong signal in all the corridors. Each chamber can have an arbitrary number of transmitters installed.

Input

The first line of the standard input contains a positive integer n ($1 \le n \le 200\,000$) that specifies the number of chambers in the mine. These chambers are numbered from 1 to n.

The following $n-1$ lines describe the mine's corridors. Each such line contains two integers a and b ($1 \le a, b \le n$, $a \ne b$), separated by a single space, which indicate that the chambers number a and b are directly linked by a corridor.

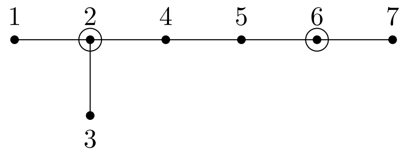
Output

The first and only line of the standard output should contain a single integer: the minimum number of transmitters that Byteasar has to deploy.

Example

For the input data:

```
7
1 2
2 3
2 4
4 5
5 6
6 7
```

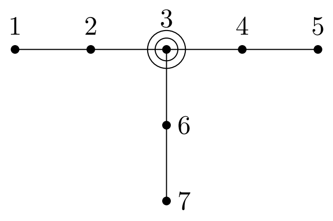


the correct result is:

```
2
```

whereas for the following input data

```
7
1 2
2 3
4 3
5 4
6 3
7 6
```



the correct result is:

```
2
```

Explanation of the example: In the first example, it is sufficient to install one transmitter in each of the chambers number 2 and 6, whereas in the second one it is sufficient to install two transmitters in the chamber number 3.

Task: NIM
Nim with a Twist

Maratona de Programação

Brazilian ICPC Summer School 2018, Day 1. Available memory: 64 MB. *29.01.2018*

The favorite pastime of Alice and Byteasar is playing *Nim*. The game starts with an arbitrary initial arrangement of tokens into heaps. The two players move alternately, where a single move consists in choosing an arbitrary heap and removing from it any positive number of tokens. The player who cannot make a move loses.

Alice has just proposed another game of Nim. To make it more interesting, this time they have decided to alter the rules slightly. Namely, Alice has distributed the m tokens into heaps of sizes $a_1, a_2, \ldots, a_n$, as usual. Then, before the game commences, Byteasar may remove some of the heaps from play. The number of removed heaps must be divisible by a predetermined number d, and at least one heap must remain. Afterwards, they are going to play a regular game of Nim, with Alice moving first.

Let k denote the number of valid subsets of heaps such that Byteasar can have them removed and be able to win the game afterward, regardless of Alice's moves. Your task is to provide the remainder of k divided by $10^9 + 7$.

Input

The first line of the standard input contains two positive integers n and d, separated by a single space, specifying the number of heaps and the divisor of the number of heaps that Byteasar may remove respectively.

The second line describes the heaps: It contains a sequence of n positive integers $a_1, a_2, \ldots, a_n$, separated by single spaces, such that a_i is the number of tokens in the i-th heap.

The conditions $n \leq 500\,000$, $d \leq 10$, $a_i \leq 1\,000\,000$ are satisfied. Moreover, the total number of tokens $m = a_1 + a_2 + \ldots + a_n$ does not exceed $10\,000\,000$.

Output

The first and only line of the standard output should contain a single integer: the number (modulo $10^9 + 7$) of different subsets of heaps that Byteasar can remove, ensuring his subsequent victory.

Example

For the input data:

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

the correct result is:

```
2
```

Explanation of the example: Byteasar can remove either 2 or 4 heaps. He is going to win only if he removes one heaps with 1 token and another one with 4 tokens; there are two sets of heaps with this property.

Task: PIO
Pawn

Brazilian ICPC Summer School 2018, Day 1. Available memory: 128 MB. *29.01.2018*

There is a pawn located at point $(0, 0)$ of an infinite grid. The pawn has n possible moves. Every move is described by a vector of integer coordinates. The pawn can perform moves in any order, but each move can be performed at most once. The vectors describing the moves can be given multiple times, and then the pawn can use each of them.

The goal is to move the pawn to a point which is located as far as possible from the starting point (in Euclidean distance). Calculate this maximal distance.

Input

The first line of the input contains one integer n ($1 \le n \le 200\,000$) specifying the number of moves. Each of the subsequent n lines specifies one move using two integers x_i, y_i ($-10^4 \le x_i, y_i \le 10^4$).

Output

Your program must write to the output one integer specifying the square of the distance from point $(0, 0)$ to the farthest point to which the pawn can be moved.

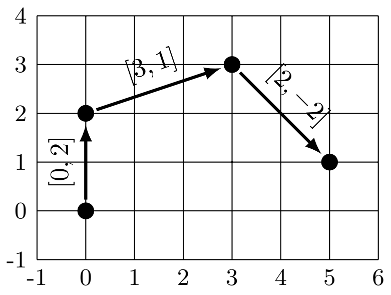
Example

For the input data:

```
5
2 -2
-2 -2
0 2
3 1
-3 1
```

the correct answer is:

```
26
```



Task: POL
Polarization

Brazilian ICPC Summer School 2018, Day 1. Available memory: 128 MB. *29.01.2018*

Everyone knew it would only be a matter of time. So what? Faced for years on, a peril becomes the every-day reality. It loses its meaning...

Today the letter of the Bitotian char Bittard to the Byteotian king Byteasar was released to the public. Bitotia requested annexation of the whole Byteotia on pain of using the Bit Polarizing Magnet (BPM). If used, the BPM would make each and every road in Byteotia unidirectional. The enemy knows only too well that this could be a fatal blow to the minimalist Byteotian infrastructure – there is a unique way between each pair of towns.

How badly can the BPM damage the Byteotian infrastructure? Determine the minimum and the maximum number of such pairs of towns that it will still be possible to travel from one of them to the other while observing the new roads orientation.

Input

The first line of the standard input gives a single integer n ($1 \leq n \leq 250\,000$), the number of towns in Byteotia. The $n-1$ lines that follow describe these roads. Each such line holds two integers, u and v ($1 \leq u < v \leq n$), which indicate that there is a direct road (still bidirectional at the moment) linking the towns number u and v.

Output

Two integers should be printed to the first and only line of the standard output. The first number should be the minimum and the second – the maximum number of pairs of towns which could remain connected (though in one direction only) after the roads are polarized.

Example

For the input data:

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

the correct result is:

```
3 5
```

and for the input data:

```
8
1 2
2 3
3 4
4 5
5 6
6 7
7 8
```

the correct result is:

```
7 28
```

Task: PRA
Important Meeting

Brazilian ICPC Summer School 2018, Day 1. Available memory: 128 MB. *29.01.2018*

Byteasar needs to organize an important meeting in his company. The company has n employees, and he needs a group of k employees to attend the meeting. Each employee has a time period, when he or she is free. You have to choose k employees, such that the time for the meeting (i.e. the time in which all these employees are free) is as long as possible.

Input

The first line of the input consists of two integers n and k ($1 \le k \le n \le 1\,000\,000$) specifying the total number of employees in the company and the number of employees needed for the meeting. The employees are numbered from 1 to n.

In the subsequent n lines there is information about the employees; i-th of these lines consists of two integers a_i and b_i ($1 \le a_i < b_i \le 10^9$) specifying that the i-th employee is free between moment a_i and moment b_i.

Output

In the first line of the output you have to write an integer specifying the maximum possible length of the meeting. You can assume that it will be possible to have a meeting of length at least 1. In the second line you have to write a sequence of k integers, separated by single spaces, which specify the numbers of employees at the meeting. If there is more than one correct answer, you can write any of them.

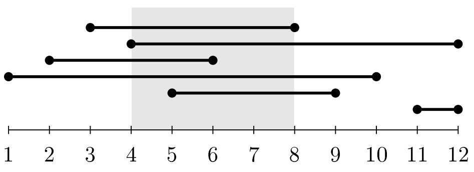
Example

For the input data:

```
6 3
3 8
4 12
2 6
1 10
5 9
11 12
```

a correct answer is:

```
4
1 2 4
```



Explanation of the example: The longest possible meeting of three employees has length 4; it could consist of employees number 1, 2 and 4.

Task: ROZ
Difference

Brazilian ICPC Summer School 2018, Day 1. Available memory: 128 MB. *29.01.2018*

A word consisting of n lower-case letters of the English alphabet (`a` to `z`) is given. We would like to choose a non-empty contiguous (i.e. one-piece) fragment of the word so as to maximise the difference in the number of occurrences of the most and the least frequent letter in the fragment. We are assuming that the least frequent letter has to occur at least once in the resulting fragment. In particular, should the fragment contain occurrences of only one letter, then the most and the least frequent letter in it coincide.

Input

The first line of the standard input holds one integer n ($1 \leq n \leq 1\,000\,000$) that denotes the length of the word. The second line holds a word consisting of lower-case letters of the English alphabet.

Output

The first and only line of the standard output is to hold a single integer, equal to the maximum difference in the number of occurrences of the most and the least frequent letter that is attained in some non-empty contiguous fragment of the input word.

Example

For the input data:

```
10
aabbaaabab
```

the correct result is:

```
3
```

Explanation of the example: The fragment that attains the difference of 3 in the number of occurrences of `a` and `b` is `aaaba`.

Task: SZA
Cloakroom

Brazilian ICPC Summer School 2018, Day 1. Available memory: 128 MB. *29.01.2018*

The annual rich citizen's reunion is taking place in Byteotia. They meet to boast about their incomes, Lebytene's shoes and other luxuries. Naturally, not all these objects of pride are carried into the banquet – coats, jackets, umbrellas and such are left in the cloakroom, and picked up upon leave.

Unfortunately for the well off, a gang of Byteotian thieves plans to break into the cloakroom and steal part of the items stored therein. At this very moment the gang's leader is examining the plans of the robbery put forward by other gang members (they are a democratic lot!). A plan typically looks as follows: the thieves break into the cloakroom at time m_j, take items worth *exactly* k_j and escape, where the whole heist takes them time s_j. The gang leader would first of all like to know which of these plans are feasible and which are not. A plan is feasible if at time m_j it is possible to collect items of total value k_j in such a way that up to the very moment $m_j + s_j$ no one shows up to retrieve any of the stolen goods (in such event, they would notify security, and the robbery would fail). In particular, if at time m_j it is impossible to select items of exact total worth k_j, then the plan is infeasible and consequently rejected. Knowing the drop off and retrieval times for each item, determine which plans are feasible and which are not. We assume that an item left in the cloakroom the moment a robbery starts can already be stolen (see the example).

Input

In the first line of the standard input there is a single integer n ($1 \leq n \leq 1000$) denoting the number of items that will be left in the cloakroom. Those items are described in the n lines that follow. Each of those lines consists of three integers c_i, a_i, and b_i ($1 \leq c_i \leq 1000$, $1 \leq a_i < b_i \leq 10^9$), separated by single spaces, that denote respectively: the item's value, the moment it is left in the cloakroom, and the moment it will be retrieved by the owner.

The next line holds an integer p ($1 \leq p \leq 1\,000\,000$), the number of plans the gang came up with. Each is specified in a separate line by three integers, m_j, k_j and s_j ($1 \leq m_j \leq 10^9$, $1 \leq k_j \leq 100\,000$, $0 \leq s_j \leq 10^9$), separated by single spaces, that denote respectively: the moment the thieves would enter the cloakroom, the value of goods they would like to steal, and the time the robbery would take them.

Output

For each plan put forward by the gang determine if it is feasible, i.e., whether it is possible to steal items of total worth exactly k_j and escape before anyone asks for their belongings. If the plan is feasible, your program should print the word `TAK` (Polish for *yes*) on the standard output, otherwise it should print `NIE` (Polish for *no*).

Example

For the input data:

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

the correct result is:

```
TAK
NIE
TAK
TAK
NIE
```

Task: WIE
Polygon

Brazilian ICPC Summer School 2018, Day 1. Available memory: 256 MB. *29.01.2018*

There are n mutually disjoint segments on the plane, each segment is parallel to one of the axes of Cartesian system. We want to construct a polygon having at most $20n$ sides parallel to the axes of the system, such that some of its n sides coincide with the given segments.

Each two subsequent sides of the polygon must be perpendicular to each other, and the sides cannot have common points (except the common ends of two subsequent sides). Exactly n sides of the polygon must coincide with some segment. The coordinates of the segments are integers, but the coordinates of the vertices of the polygon can be fractional.

Input

The first line of the input contains one integer n ($1 \le n \le 100\,000$) specifying the number of segments.

Subsequent n lines specifies one segment each. A segment is specified by four integers x_1, y_1, x_2, y_2 ($-10^8 \le x_1, y_1, x_2, y_2 \le 10^8$), which means that the segment is from (x_1, y_1) to (x_2, y_2). Each segment has non-zero length and it is either horizontal or vertical. The segments do not have common points.

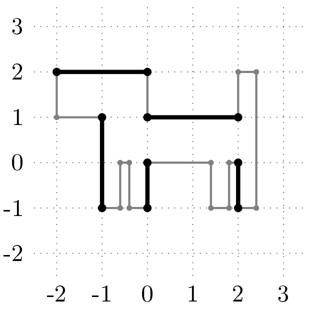
Output

If it is not possible to construct a polygon, you have to write a word `NIE` (Polish for *no*) in the only line of the output. Otherwise in the first line of the output you have to write an integer m ($m \le 20n$), specifying the number of sides of the polygon. In the subsequent m lines you have to write vertices of the polygon in clockwise or counter-clockwise order; i-th of these lines should contain two real numbers x, y ($-10^9 \le x, y \le 10^9$) specifying that the coordinates of the i-th vertex of the polygon are (x, y). Each of these numbers can have at most seven digits after the decimal dot.

Example

For the input data:

```
5
-1 1 -1 -1
0 0 0 -1
0 1 2 1
2 -1 2 0
0 2 -2 2
```



a correct answer could be:

```
22
-1 1
-1 -1
-0.6 -1
-0.6 0
-0.4 0
-0.4 -1
0 -1
0 0
1.4 0
1.4 -1
1.8 -1
1.8 0
2 0
2 -1
2.4 -1
2.4 2
2 2
2 1
0 1
0 2
-2 2
-2 1
```

Task: WIL
Trous de loup

Brazilian ICPC Summer School 2018, Day 1. Available memory: 128 MB. *29.01.2018*

The King of Byteotia, Byteasar III the Bold, is planning a raid on enemy's castle. On three sides, the castle is surrounded by impenetrable moat, leaving Byteasar no other option than to lay siege to the fourth side of the castle. However, it turns out that this side is not completely unprotected either: the royal scouts have reported that there are deep trous de loup (think of them as big holes in the ground) along this wall. Byteasar would like to attack a contiguous segment of the wall as long as possible. To this end, some of the trous de loup will have to be disabled. The wise king has decided to fill up some of them with sand, and to cover some with the Great Plank.

Along the wall, there are n trous de loup. King Byteasar's troops have p bags of sand. To fill up the i-th trou de loup, w_i of the bags of sand are required. Moreover, the Great Plank can cover up to d successive trous de loup.

Help Byteasar in identifying the longest segment of the wall that his troops may attack if they utilize their resources (i.e., the bags of sand and the Great Plank) optimally. In other words, determine what is the maximum number of successive trous de loup that can be disabled.

Input

The first line of the standard input contains three integers, n, p, and d ($1 \leq d \leq n \leq 2\,000\,000$, $0 \leq p \leq 10^{16}$), separated by single spaces, which specify the numbers of trous de loup and of bags of sand, and the length of the Great Plank respectively.

The next lines describes the trous de loup: it contains a sequence of n integers, $w_1, w_2, \ldots, w_n$ ($1 \leq w_i \leq 10^9$) separated by single spaces; w_i is the number of bags of sand required to fill up the i-th trou de loup.

Output

The first and only line of the standard output should contain a single integer: the length of the longest contiguous segment of the wall that Byteasar's troops can attack.

Example

For the input data:

```
9 7 2
3 4 1 9 4 1 7 1 3
```

the correct result is:

```
5
```

Explanation: The trous de loup number 2, 3, and 6 can be filled up with sand (using 6 out of the 7 bags available), whereas the ones numbered 4 and 5 can be covered with the Great Plank. This way, five successive trous de loup (the ones numbered 2 to 6) will be disabled.

Task: ZAP
Panini

Brazilian ICPC Summer School 2018, Day 1. Available memory: 256 MB. *29.01.2018*

Byteasar runs a world-famous panini store, known for its products quality and unique flavor. Over the years, he attained a stable market position and a group of patrons.

Everyday, k customers come to his store, the i-th one arriving at time t_i. Every customer orders a single panino. Byteasar would love to serve each of his loyal customers immediately, but unfortunately this is not possible: He can grill at most z panini in a single batch, and the grilling process takes exactly d units of time. Needles to say, the grilling cannot be paused, stopped, or interfered with in any way (e.g., to add or remove a panino), as this would spoil the amazing taste. Thus, Byteasar will settle for minimizing the total service time of his patrons, i.e., the total time they have to wait for their orders. Byteasar knows his customers so well that he knows their arrival times and favorite panini, so he may start grilling them in advance, before a customer arrives. However, no customer's order may be ready before they arrive, as no one likes a cold panino! Byteasar shows up at his store at time 0.

What is the minimum total service / waiting time of the customers?

Input

The first line of the standard input contains three positive integers k, z and d ($1 \leq z \leq k \leq 3000$, $1 \leq d \leq 1\,000\,000$) which specify the number of customers, the grill capacity, and the grilling duration respectively.

In the second line, there is a sequence of k integers $t_1, t_2, \ldots, t_k$ ($0 \leq t_1 \leq t_2 \leq \ldots \leq t_k \leq 1\,000\,000$); the number t_i is the arrival time of the i-th customer.

Output

Exactly one integer should be printed to the standard output – the total service / waiting time of the customers in the optimum grilling schedule.

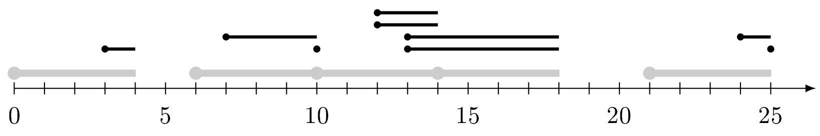
Example

For the input data:

```
9 2 4
3 7 10 12 12 13 13 24 25
```

the correct result is:

```
19
```



Explanation for the example: In the optimal grilling schedule (depicted in the figure), Byteasar starts grilling at times 0, 6, 10, 14 and 21. The first time he grills but a single panino, grilling two at once ever since. The grilling time is marked in gray, whereas the waiting time in black.