Task: FIL
Film Editor

Maratona de Programação, Day 3. Available memory: 256 MB. *28.01.2015*

Limonia is a new dynamically developing city. Andrew Citrus (the Mayor of Limonia) decided that it should be promoted in order to increase the number of tourists visiting it in the future. For this purpose, he hired a film crew whose task was to make a film depicting the most interesting sides of Limonia. The filmmakers carried out their duty perfectly and prepared a film consisting of n shots depicting many interesting aspects of Limonia, e.g. the picturesque pen of Puffnootsy the wombat or the well-developed chain of pizzerias. However, it displeased Citrus that not the entire film had a dynamic character compatible with the pace of development of Limonia. Some of the shots were dynamic indeed, but more static shots also appeared, for example conversations with very important persons.

As city promotion films do not usually have strictly imposed chronology, the Major decided to edit the shots on his own. He cares about maximising the length of the longest dynamic scene. He would not like to introduce too much chaos though, so he decided to cut the film reel using at most k cuts and edit the resulting fragments in order of his choice (of course he cannot reverse a fragment back-to-front). Help the Major and tell him how long dynamic scene he can edit!

A scene is a sequence of consecutive shots and its length is the number of shots it consists of. A scene is called dynamic if it consists of dynamic shots only.

Input

In the first line of the input there is one integer t ($1 \leq t \leq 10$) specifying the number of test cases. In every of the following t lines a description of one test case can be found. It consists of an integer k ($0 \leq k \leq 100\,000$) specifying the maximum number of cuts which may be performed by Andrew Citrus and a n-letter word ($1 \leq n \leq 100\,000$) consisting of letters `D` and `S`: the i-th letter stands for the type of the i-th shot in the film (the letter `D` stands for a dynamic shot, and the letter `S` stands for a static shot).

Output

The output should consist of t lines containing answers to the consecutive test cases from the input. The answer to a test case is an integer meaning the length of the longest dynamic scene which can be edited by the Major.

Example

For the input data:

```
2
2 DDSDSSD
3 DSDSDSD
```

the correct result is:

```
4
3
```

Explanation of the example: If we cut the first film using two cuts to three pieces of lengths respectively 3, 1 and 3 shots, and next we edit them in the reverse order, then all 4 dynamic shots will be placed in one dynamic scene.

Task: BOH
The Hero

Maratona de Programação, Day 3. Available memory: 256 MB. *28.01.2015*

A new computer game has just been released and presently is available in Byteotian supermarkets, In the game we play the role of a brave hero Bithor, whose task is to destroy the n monsters living in the Byteburg dungeons. For simplicity, the monsters will be numbered from 1 to n.

During the encounter with the monster numbered i Bithor suffers injuries that cost him d_i life points. The monster's task is to defend a chest containing the healing potion, capable of restoring a_i of Bithor's life points, providing he wins an engagement with the monster.

Bithor defeats monsters without difficulty, however, he absolutely cannot allow the number of his life points to drop to zero (or below) at any time. Is it possible that Bithor fights enemies in such an order, that defeats all of them?

Input

The first line contains two integers n and z ($1 \leq n, z \leq 100\,000$), specifying the number of monsters and the initial number of Bithor's life points. The next n lines contain descriptions of monsters: i-th of these lines contains two integers d_i and a_i ($0 \leq d_i, a_i \leq 100\,000$), specifying the damage inflicted by a monster number i and the power of the healing potion which could be consumed after defeating the monster.

Output

The first line of output should contain one word `TAK` (Polish for *yes*) or `NIE` (Polish for *no*), depending on whether Bithor is able to defeat all the monsters. In case defeating all the monsters is possible, a second line should be produced containing a string of n pairwise distinct integers ranging from 1 to n, separated by single spaces. This string should describe an exemplary fight sequence, and its consecutive elements should correspond to the numbers of sequentially conquered monsters. In case there is more than one correct answer, your program should output any of them.

Example

For the input data:

```
3 5
3 1
4 8
8 3
```

the correct result is:

```
TAK
2 3 1
```

Task: KUG
The Juggler

Maratona de Programação, Day 3. Available memory: 256 MB. *28.01.2015*

Byteson earns a living as a juggler at a fair in Byteland. He invites passers-by to a specific game. There is a row of n cups numbered $1, 2, \ldots, n$ on the table. Rubber marbles are are hidden under some of them. In case a player accurately guesses under which cups the marbles are hidden, he win a big teddy bear. Byteson is offering hints to players for a fee. For c_{ij} bytecents Byteson is willing to reveal *the parity of the number* of marbles hidden under the cups numbered $i, i+1, \ldots, j$.

Byteasar visited the fair with Byteatrix — prettiest girl in Byteland. He is determined to to win a teddy bear for her. Definitely he hates to be embarrassed and he is certain that guesswork is not the right way to go. He intends to pay for hints until all gathered information would enable him to determine, with certainty, which cups have marbles underneath.

Knowing the prices of all the possible hints, he is wondering now, what would be the maximum cost. To be precise, he would like to know the least such number k, that there is a strategy of asking questions such that, regardless of the answers Byteson is providing, allows to determine the marbles positioning for at most k bytecents.

Input

The first line of the input contains one integer n ($1 \leq n \leq 2000$), specifying the number of cups. This is followed by a description of the cost of hints about the individual intervals. The $(i+1)$-th line of input contains $n+1-i$ integers (for $1 \leq i \leq n$), specifying the costs of individual hints.

The hints cost c_{ij} ($1 \leq i \leq j \leq n$, $1 \leq c_{ij} \leq 10^9$) concerning the range from i-th to j-th cup (inclusive), appears at the input as $(j+1-i)$-th number of the $(i+1)$-th line.

Output

Your program should output one integer — the maximum cost in bytecents of determining the positioning of the marbles for optimal stategy of asking questions.

Example

For the input data:

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

the correct result is:

```
7
```

Task: PAR
The Car Park

Maratona de Programação, Day 3. Available memory: 256 MB. *28.01.2015*

Have you ever wondered what being a car park employee is really all about? On the surface it does not look like a difficult job: customers leave their cars that need to be parked. Simple? It might seem so, but the reality is more complicated. The main problem of our car park guy is his boss.

One day the boss told his poor employee to move all the cars in the car park according to his whim. Our guy, looking at the piece of paper from his boss, and at a piece of paper with the current car arrangement, is wondering whether it is possible at all. Help him!

Let us represent the car park as a rectangle (positioned on a plane) with the height of w. For convenience, let us set a rectangular coordinate system with axes parallel to the sides of the rectangle. The origin of the coordinate system is located in the lower left corner of the rectangle. The car park is wide enough, so it is convenient to assume, that the right side of the rectangle is infinitely far away.

For simplicity, we also consider cars to be of a rectangular shape with sides parallel to the coordinate axes. However, cars tend to be of various sizes, and therefore the corresponding rectangles may have different sizes as well. The arrangement of the cars in the car park is correct if all the cars are located within the the car park and no part of the car park is occupied by two vehicles at the same time. We allow, however, that the edges of the rectangles, representing the cars, to overlap.

Our friend is an experienced car park employee, his seniority undisputed, so cars can be moved by him freely in any direction (formally speaking: car park worker can always move any car by any vector), providing cars are not bumped together. They, however, can not be rotated.

Your task is to determine whether you can rearrange the cars from their current positions to the positions specified by the boss, without damaging any vehicle.

Input

The first line of input contains the number of test cases t ($1 \leq t \leq 20$), which are described in the rest of the input. Description of each test case starts with a line containing two integers n and w ($1 \leq n \leq 50\,000$, $1 \leq w \leq 10^9$) specifying the number of cars in the car park and the height of the rectangle representing the car park.

The next n lines contain a description of the initial arrangement of the vehicles: i-th of these lines contains four integers x_1, y_1, x_2 and y_2 ($0 \leq x_1, x_2 \leq 10^9$, $0 \leq y_1, y_2 \leq w$), which describe a rectangle with opposite vertices located at the points (x_1, y_1) and (x_2, y_2), corresponding to i-th car at the car park. All rectangle areas are described by a positive number.

The next n lines contain a description of the target vehicle arrangement, given in an analogous format. Cars in both descriptions are given in the same order (i-th car from the initial arrangement description corresponds to the i-th car from the targeted car arrangement). The same car can be described in the target arrangement by other vertices, than the ones from the initial arrangement. You may assume that both descriptions are correct.

Output

The output should produce t rows: i-th row should contain one word `TAK` (Polish for *yes*) or `NO` (Polish for *no*) depending on whether for the i-th test case it is possible to rearrange the cars in accordance to the requirements of the boss, or not.

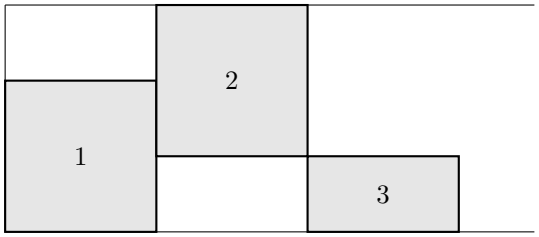
Example

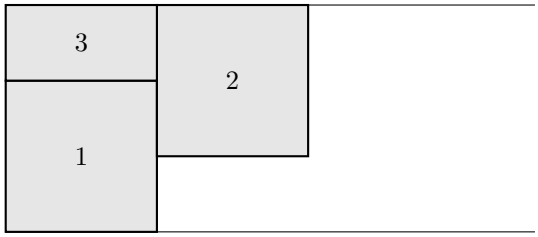
For the input data:

```
2
3 3
0 0 2 2
2 1 4 3
4 0 6 1
0 0 2 2
2 1 4 3
0 2 2 3
3 3
0 0 2 2
2 1 4 3
4 0 6 1
2 1 4 3
0 0 2 2
4 0 6 1
```

the correct result is:

```
TAK
NIE
```





Explanation of the example: The figure shows the first test case from the example. The left part shows the initial setting of the cars, and the right part shows a destination arrangement. In order to position car number 3 at the correct location, car number 2 should be moved down or to the right. In the second test case we would have to swap cars number 1 and 2, which is not possible.

Task: KOR
Woodworms

Maratona de Programação, Day 3. Available memory: 256 MB. *28.01.2015*

Two woodworms decided to eat an old wooden fence. The fence consists of n planks, which are not necessarily the same height. Woodworms heard from friendly termites, that nothing makes a meal more pleasant, than a little healthy competition. Therefore they decided to play a game and eat fence planks in turns. A woodworm, taking its turn, can eat one of the planks from the end of the fence or both of them at once. Knowing that, each woodworm chooses the planks so that the total of their heights, eaten during the whole game, to be as large as possible, calculate how much wood would be eaten by each of them.

Input

The first line of input contains an integer n ($1 \leq n \leq 1\,000\,000$), specifying the number of planks in the fence. The second line contains a sequence of n integers $h_1, h_2, \ldots, h_n$ ($1 \leq h_i \leq 1\,000\,000\,000$), describing the lengths of consecutive fence planks.

Output

The first and only line of output should contain two integers. The first one denotes the total fence planks heights, which are the nourishment of the woodworm beginning the game, and the second, how much wood would fall in the share of its opponent.

Example

For the input data:

```
4
5 2 9 3
```

the correct result is:

```
14 5
```

Explanation of the example: In its first move, the first woodworm can choose a plank of height 5, of height 3, or both at once. The optimal move is eating a plank of height 5. The opponent then eats planks of heights 2 and 3.

Task: SKO
Matchings

Maratona de Programação, Day 3. Available memory: 32 or 256 MB. *28.01.2015*

In an undirected graph, a matching is a subset of the edges of the graph such that each vertex of the graph is adjacent to at most one of the selected edges. The maximum matching is a matching of maximum possible cardinality.

You are given a tree with n nodes. Your task is to find the size of the maximum matching and the number of maximum matchings (the latter one modulo m).

Input

The first line of the input contains an integer n that specifies the number of nodes of the tree ($1 \leq n \leq 1\,500\,000$). The nodes are numbered 1 through n. The following $n-1$ lines contain a description of the tree edges. Each of the lines contains two integers a and b that represent an edge connecting the nodes a and b ($1 \leq a, b \leq n$). The last line of the input contains an integer m ($1 \leq m \leq 10^9$).

Output

The first line of the output should contain the cardinality of the maximum matching in the tree. The second line should contain the number of maximum matchings modulo m.

Example

For the input data:

```
5
1 2
3 2
4 5
1 4
17
```

the correct result is:

```
2
3
```

Task: NEO
Neon

Maratona de Programação, Day 3. Available memory: 256 MB. *28.01.2015*

Byteasar cannot sleep, because of the big neon sign which has been recently mounted opposite to his window. On the neon there are two words which advertise a nearby cafe. Due to lack of sleep, strange ideas come to Byteasar. He wonders how many possibilities there are to turn off some letters on the neon sign in such a way, that the letters which are still on are arranged in both words in the same non-empty subsequence. He also wonders how many such different subsequences one can get.

Input

In the first line of the input there is the first word of the neon sign, and in the second line there is the second word. Both words are non-empty, consist of small letters of English alphabet and of length no greater that 10 000.

Output

On the output you have two write one line consisting of two integers separated by a single space. The should denote respectively: the number of ways to turn off letters in the neon sign and the number of possible subsequences. Both numbers should be written modulo $10^9 + 7$.

Example

For the input data:

```
supermarket
stokrotka
```

the correct result is:

```
27 17
```

Explanation of the example: We can get 17 different subsequences (`skt`, `sra`, `srk`, `srt` and their shorter subsequences) in 27 ways (e.g. we can obtain `srk` in two different ways).

Task: PRI
Prime numbers

Maratona de Programação, Day 3. Available memory: 32 MB. *28.01.2015*

Professor Bytheon wants to explore the fascinating world of prime numbers. He asked you to write a program which will calculate some values for him. The professor would like to know what is the sum of prime numbers in the range $[a, b]$ if we sum every m-th prime number from this range.

Input

In the first line of the input there are three integers a, b and m describing professor's question ($1 \leq a \leq b \leq 10^8$, $1 \leq m \leq 10^8$).

Output

In the only line of the output you have to write the answer to the question.

Example

For the input data:

```
3 18 3
```

the correct result is:

```
14
```

Explanation of the example: In the range $[3, 18]$ there are six prime numbers: 3, 5, 7, 11, 13 and 17. Summing every third of them we get $3 + 11 = 14$.