Problem A. AMPPZ in the times of disease

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	1024 mebibytes

Organizing AMPPZ in the times of pandemic is quite a challenge. As the Head Judge of Social Distancing, it is your job to ensure that participants keep a safe distance from each other. Since students from a single university are practically a family, you're mostly concerned with the distance between pairs of students from different universities. Intuitively, you want the students from the same university to form a tight group, keeping a safe distance from all other such groups.

To formalize your intuition, you came up with the following rule: let A denote the largest euclidean distance (standard distance on the plane) between two of students from the same university, and let B denote the smallest euclidean distance between two students from different universities. Then your rule states that $A < B$ must hold.

All your guests accepted the guidelines, and upheld them throughout the event. However, there's a catch: after the competition ended, you were asked to prove that the social distancing rules were indeed respected. Everybody is already gone, and the only thing left is to try to use one of the group photos as a proof... problem is, you don't know which contestants were affiliated with which university! But **since you know, that the social distancing rule was upheld**, maybe you can recover the division into universities?

Knowing the positions of all students in the picture (described as points on the plane: the group photo was taken from above using a drone, as this was the angle from which the contestants looked the best.) and the number of universities, divide students into universities in a way that respects your social distancing rule. **Every university has to have at least one student; moreover, you can assume that the solution always exists**.

Input

The first line of input contains the number of test cases z ($1 \le z \le 100\,000$). The descriptions of the test cases follow.

The first line of a test case contains two integers n, k ($2 \le n \le 2\,000\,000$, $2 \le k \le \min(n, 20)$), denoting the number of students and the number of universities, respectively.

The next n lines describe the positions of the students. Each contains two integers x_i, y_i ($0 \le x_i, y_i < 10^9$), denoting the coordinates of the i-th student. **No two students stand in exactly the same place.**

The total number of students across all test cases does not exceed 10^7.

Output

For every test case, output n integers $c_1, \cdots, c_n$ ($1 \le c_i \le k$): a division of students into universities that satisfies the social distancing rule. If there are multiple solutions, you can output any of them.

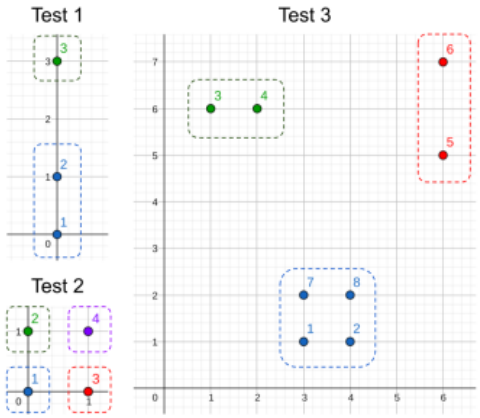
Example

standard input	standard output
3 3 2 0 0 0 1 0 3 4 4 0 0 0 1 1 0 1 1 8 3 3 1 4 1 1 6 2 6 6 5 6 7 3 2 4 2	1 1 2 4 1 3 2 2 2 1 1 3 3 2 2

Note

Blank lines in the example input were added for readability. They are not present in the real input files.

Below we show the sample test cases together with possible correct answers.



Problem B. Babushka and her pierogi

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	1024 mebibytes

Babushka Bajtmiła is throwing a party! And the main dish served will be her famous *pierogi*.

There will be n plates available during the party, and Babushka plans to put exactly p_i pierogi on the i-th plate (all the values p_i are distinct). Though the task seemed too heavy for an old lady, Bajtmiła has stood up to the challenge and almost instantly prepared all of the pierogi needed, divided among n plates with $(p_1, \ldots, p_n)$ pierogi. Next, Bajtmiła has distributed the plates among the plates. However, she soon realized that while she got the numbers right, she messed up the order of the plates.

Bajtmiła is quite tired and is only willing to perform one type of operation: she can choose two plates numbered i and j, and swap the amounts of pierogi on each plate. In other words, if there are x pierogi at the plate i, and y pierogi at the plate j, then after this operation there will be y pierogi at the plate i, and x pierogi at plate j. Such an operation takes exactly $|x - y| + C$ seconds to perform – C seconds for finding a proper spoon, and 1 second for each of pierogi moved.

The party is about to start very soon! Now, Bajtmiła won't allow you to touch anything in the kitchen, but she has put her trust in your algorithmic skills. She asked you to find a sequence of operations restoring the desired order of numbers, and must do it in the shortest time possible. Can you help Bajtmiła?

Input

The first line of input contains the number of test cases z ($1 \le z \le 1000$). The descriptions of the test cases follow.

The first line of each test case consists of two numbers n and C ($1 \le n \le 200\,000, 1 \le C \le 10^9$) with their meaning described in the statement above.

Next n lines describe consecutive plates. The i-th line contains two numbers a_i and p_i ($1 \le a_i, p_i \le 10^9$) indicating the current and the desired amount of pierogi on i-th plate respectively.

In each test case, numbers a_i are distinct. Furthermore, the sets $\{a_1, a_2, ..., a_n\}$ and $\{p_1, p_2, ..., p_n\}$ are the same.

The sum over values n in all test cases does not exceed 1 000 000.

Output

For each test case, your output must match the following description:

In the first line, print two integers S and K – the total time and the number of operations in your solution respectively.

Next K lines of your output should describe your solution. In the k-th line print two numbers x_k and y_k, indicating that the k-th operation in your solution swaps the amounts of pierogi at x_k-th and y_k-th plate.

After all operations from your solution, the i-th plate must contain exactly p_i pierogi.

Examples

standard input	standard output
1 4 2 2 4 3 2 1 1 4 3	6 2 2 1 4 1

Note

A sequence $(2, 3, 1, 4)$ should become the sequence $(4, 2, 1, 3)$. We first perform the operation on the first two plates, obtaining the sequence $(3, 2, 1, 4)$ at the cost $|3 - 2| + 2 = 3$. The second and final operation swaps the pierogis from the first and fourth plate, achieving the desired sequence $(4, 2, 1, 3)$. The cost of this operation is $|3 - 4| + 2 = 3$, and the total cost is 6, which is the minimal possible.

Problem C. Cake

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

Once upon a time Babushka Bajtmiła (known for the exception pierogi from problem B) baked a cake. She cut it into $2n$ rectangular pieces (two rows containing n pieces each) and put colour icing on the top of each piece. She looked at her masterpiece and was taken aback: the colour scheme looked absolutely awful.

Bajtmiła decided that she needs to change the colour configuration to something better. Unfortunately, changing the locations of the pieces one by one is out of question: any attempt to take out a single piece out of the cake would inevitably make its sides chipped. Babushka Bajtmiła would never serve her guests a cake which appears raggedly cut!

Fortunately, Bajtmiła possesses a rectangular spatula which can fit exactly four pieces (in two rows containing two pieces each). She can therefore carefully take four such pieces out, rotate the spatula and insert them back from the opposite side. We can formally describe this operation as taking a 2 x 2 square and rotating it by 180 degrees.

Remembering your quick and efficient help in problem B, Babushka knew exactly what to do: she asked you to determine the minimum possible number of operations that would turn her initial frosting configuration into something nice. Of course, you did what any decent person would do in your place: you said you won't be able to help because the problem is not well-defined. Bajtmiła expected that: she went to the whiteboard and draw one specific *nice* configuration and asked you to find the minimal number of operations to change her initial frosting configuration into this specific one.

It might have also happened that the granny (tired from moving all the pierogi in problem B) was mistaken and it is actually impossible to obtain the desired configuration. In such case you also need to notify her as soon as possible!

Input

The first line of input contains the number of test cases z ($1 \le z \le 10\,000$). The descriptions of the test cases follow.

The first line of a test case contains an integer n ($2 \le n \le 500\,000$). The following two lines of input describe two rows of the initial frosting configuration. Each of them contains n integers from the range $[1, 10^9]$ separated by single spaces — identifiers of frosting colours on the subsequent pieces of the cake. Note that a colour may appear more than once.

The next two lines describe the final configuration in the same format.

The total sum of n over all the test cases does not exceed $2\,000\,000$.

Output

For each test case output a single integer — the minimum possible number of operations needed to achieve the final configuration. If it is not possible to achieve the desired outcome, print -1.

Example

standard input	standard output
2 4 1 2 3 2 4 3 1 3 3 2 1 1 2 3 4 3 2 1 2 3 4 3 4 1 2	3 -1

Note

In the first test case the initial configuration is

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

We can achieve the desired configuration in three steps:

1. rotating the leftmost square,

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

2. rotating the rightmost square,

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

3. rotating the middle square.

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

It is not possible to achieve the desired configuration in the second test case.

Problem D. Divided Mechanism

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

The Bytegate machine is Bajtazar's newest invention. It consists of two parts, which – to quote enthusiastic Bajtazar – "are simply impossible to split apart!"... or so he thought. His two-year-old son Bajtuś is just about to prove him wrong.

The initial state of mechanism can be described as an $n \times m$ array, filled with characters `A`, `B` and dots. Characters `A` correspond to the first part of mechanism, characters `B` – the second part of mechanism, and dots mean empty spaces:

```
+-+-+-+-+-+
|B|B|A|A|.|
+-+-+-+-+-+
|.|B|B|A|A|
+-+-+-+-+-+
|A|.|B|B|A|
+-+-+-+-+-+
|A|.|.|B|A|
+-+-+-+-+-+
|A|A|A|A|A|
+-+-+-+-+-+
```

Since the mechanism consists of two parts, the array contains at least one character `A` and at least one character `B`. Furthermore, both parts of the mechanism are connected, i.e. for any two A-cells there exists some path connecting these cells, passing only through other A-cells, with any two consecutive cells on this path sharing a common edge. The part B is connected in the same way.

Bajtuś plays with the machine by pulling part B in various directions. His play can be described as a sequence of q letters `N`, `S`, `E`, `W` – the directions of the consecutive moves (North, South, East and West, respectively). Each time, Bajtuś is pulling the component B in the chosen direction until any further pulling in that direction would cause the parts of the mechanism to overlap. If Bajtuś could continue that movement indefinitely, we say that the two parts have been split apart. It does not mean that Bajtuś stopped messing with the mechanism at that moment – nonetheless, once split, the mechanism remains split forever. Determine if Bajtuś will actually split the mechanism during his play.

Input

The first line of input contains the number of test cases z ($1 \le z \le 1\,000$). The descriptions of the test cases follow.

The first line of each test case contains three integers n, m, q ($1 \le n, m \le 10$, $1 \le q \le 100$). The next n lines describe the initial state of the mechanism. Each of these lines is a string of length m consisting of characters `A`, `B` and `.`. Both mechanism components are non-empty and connected.

The last line of each test case contains q characters belonging to the set $\{N, S, E, W\}$, describing Bajtuś's moves.

Output

For each test case print `TAK` if Bajtuś has split the mechanism. Otherwise, print a single word `NIE`.

Example

standard input	standard output
3 5 5 3 BBAA. .BBAA A.BBA A..BA AAAAA WNW 5 5 7 BBAA. .BBAA A.BBA A..BA AAAAA WNWNSEN 6 5 3 ..... .AAA. .A.A. .AB.. .A.A. .AAA. SNE	NIE TAK NIE

Note

The final states of mechanism in the first and the third test case are:

```
+-+-+-+-+-+-+-+-+-+
|B|B|.|.|.|.|.|.|.|
+-+-+-+-+-+-+-+-+-+
|.|B|B|.|A|A|.|.|.|
+-+-+-+-+-+-+-+-+-+
|.|.|B|B|.|A|A|.|.|
+-+-+-+-+-+-+-+-+-+
|.|.|A|B|.|.|A|.|.|
+-+-+-+-+-+-+-+-+-+
|.|.|A|.|.|.|A|.|.|
+-+-+-+-+-+-+-+-+-+
|.|.|A|A|A|A|A|.|.|
+-+-+-+-+-+-+-+-+-+
```

```
+-+-+-+
|A|A|A|
+-+-+-+
|A|B|A|
+-+-+-+
|A|.|.|
+-+-+-+
|A|.|A|
+-+-+-+
|A|A|A|
```

```
+-+-+-+
```

In the second test case Bajtuś splits the mechanism in the fourth step.

Problem E. Epidemic

Input file:	*standard input*
Output file:	*standard output*
Time limit:	9 seconds
Memory limit:	1024 mebibytes

Byteland has just shut its borders, the reason being a new strain of bytebacterium detected in the neighbouring countries. Research shows that the new strain is not only highly contagious, but also does not trigger any response of the Bytelanders' immune systems, so any person once infected will remain so – and will continue to infect others – lifelong (or, at the least, until an effective cure is discovered).

To contain the spread of bytebacterium, the Bytelandish government introduced far-reaching restrictions and launched the National Invigilation System, allowing to monitor all social interactions in the country. It was announced that the restrictions will only get lifted once it becomes certain that nobody, except for people who have been put under quarantine, is infected with bytebacterium. As the country's most renowned data science expert, you were entrusted with analyzing data from the invigilation system and determining when the restrictions can indeed get lifted.

There are n people in Byteland and, initially, any one of them can either be healthy or infected with bytebacterium. After the borders get shut, k events take place one after another, of the following types:

- You learn from the invigilation system that some group of people meets. If any of them is infected, all of them become infected as well (and will remain so indefinitely). Such a social contact is the only possible way to catch the disease (contrary to initial press reports, becoming sick due to touching infected surfaces is actually impossible).
- A given person performs bytebacterium test and the result is negative.
- A given person performs bytebacterium test and the result is positive. Such a person is immediately put under quarantine and will not participate in any social contacts from then on (although it might happen that another test is later still performed to them). You might wonder why anyone would perform a bytebacterium test on such a person, when it is clear that its result must be positive. The author of this problem has issued a relevant enquiry to Byteland's Ministry of Health, but got notified that the statutory time for response has been prolonged by three months due to the complex nature of the case at hand.
- You receive a query from the health minister to determine if the restrictions can already be abolished, i.e. whether all the information gained so far makes it possible to prove that nobody – except for the quarantined people – can be infected. If any individual might still possibly be infected, you need to provide an example of such a person, with accordance to the ministry's guidelines (described in the *Output* section).

Your solution must implement an **online** algorithm – that is, you need to be able to answer each query right after you receive it, without reading any later sections of the input.

Input

Correct interpretation of the input data will depend on the current value of the variable **shift**. At the beginning of each test case, the variable **shift** is initialized as 0, while its later values will depend on the answers returned by your program. Such input format is meant to ensure that your implementation answers each query immediately after reading it.

The decoding function is defined as follows:

```
decode (p) = ((p - 1 + shift) mod n) + 1,
```

where p is an integer satisfying $1 \leq p \leq n$, while `mod` is the operation of taking the remainder of integer division.

The first line of input contains the number of test cases z ($1 \le z \le 1\,000$). The descriptions of the test cases follow.

The first line contains two integers n and k ($1 \le n \le 500\,000$, $1 \le k \le 1\,000\,000$), denoting the number of people and the number of events respectively. People are numbered from 1 to n.

The next k lines describe the consecutive events, each taking one of the following forms:

- Letter `K` and an integer c ($2 \le c \le n$), followed by c distinct integers $p_1, \ldots, p_c$ ($1 \le p_i \le n$) – a social contact in which c people with identifiers $decode(p_1), \ldots, decode(p_c)$ participate.
- Letter `N` followed by an integer p ($1 \le p \le n$) – person number $decode(p)$ takes a bytebacterium test which gives the negative result.
- Letter `P` followed by an integer p ($1 \le p \le n$) – person number $decode(p)$ takes a bytebacterium test which gives the positive result. This person is immediately put under quarantine and will not participate in any social contacts from that moment on (while any further test performed on this person will inevitably give the positive result as well).
- Letter `Q` followed by an integer p ($1 \le p \le n$) – the ministry's inquiry with the *starting value* (see *Output* section) equal to $decode(p)$.

The sums of values of n and k over all test cases do not exceed 500 000 and 1 000 000 respectively. The sum of values of c over all queries of all test cases does not exceed 1 000 000.

Output

For each test case, output one line per each query (the type `Q` event) appearing in this test case.

If, at the moment of the i-th query, it can be proven that it is impossible for anybody (except for the quarantined individuals) to be infected, the i-th line should contain the word `TAK` (it might happen that at some point all Bytelanders are under quarantine. Of course, in such case this statement becomes trivially true and your program should print `TAK`). In such case, the variable **shift** changes its value to 0.

Otherwise, the i-th line should contain the word `NIE` followed by a single integer. If $decode(p)$ is the **starting value** of the query, the integer should be the identifier of the *first person in the sequence* $(decode(p), decode(p)+1, \ldots, n, 1, 2, \ldots, decode(p)-1)$ who might possibly be infected with bytebacterium but *is not quarantined*. This number becomes the new value of the variable **shift**.

Note that:

- The variable **shift** does not change its value on the events of type `K`, `N` or `P`.
- The positive and negative test results always describe a believable scenario. That is, for each test case there exists at least one set of initially infected people, for which the event sequence described on the input does not contain any contradictions.

Example

standard input	standard output
1 6 14 K 3 3 4 5 K 2 6 5 N 3 Q 3 P 1 K 2 6 2 P 6 Q 4 P 6 K 2 1 3 N 3 Q 4 N 2 Q 1	NIE 5 NIE 1 TAK TAK

Note

After decoding, the input shown above looks as follows:

```
1
6 14
K 3 3 4 5
K 2 6 5
N 3
Q 3
P 6
K 2 5 1
P 5
Q 3
P 1
K 2 2 4
N 4
Q 5
N 2
Q 1
```

Note

- Before the first query, individuals 3, 4 and 5 meet, then individuals 5 and 6 meet, then person 3 gets a negative test result. Upon receiving the first query, we can – using all the information collected so far – derive the following conclusions (beginning at the query's *starting value* 3). Person 3 must be healthy (he just got a negative test result). Person 4 must also be healthy (she could not have been infected at the time of the first meeting, because otherwise she would have infected individuals 3 and 5, which would yield a contradiction with the negative test result later obtained by person 3; since then, person 4 has had no possibility of getting infected, so she must be healthy at the moment). Person 5 might possibly be infected (he must have been healthy when he met with 3 and 4, but he later met with person 6 whose health status cannot be derived from the available information). The answer is therefore `NIE 5` and *shift* changes its value to 5.

- After the first query, person 6 gets a positive test result, then individuals 1 and 5 meet, then person 5 gets a positive test result. Upon receiving the second query, we can – using all the information collected so far – derive the following conclusions (beginning at the query's *starting value* 3). 3 and 4 must be healthy. 5 and 6 are under quarantine. Person 1 might be infected (actually, we can even be sure that she *must be* infected, but your program does not need to distinguish between these two cases). The answer is therefore `NIE 1` and *shift* changes its value to 1.
- Afterwards, person 1 gets a positive test result, then individuals 2 and 4 meet, then person 4 gets a negative test result. Upon receiving the second query, we can derive the following conclusions (beginning at the starting value 5). Individuals 5, 6 and 1 are under quarantine. Person 2 must be healthy because of the negative test result obtained by person 4. Individuals 3 and 4 must also be healthy. The answer is therefore `TAK`, because it can be proven that all non-quarantined people (2, 3 and 4) must be healthy. The variable **shift** changes its value to 0.
- After the third query, person 2 gets a negative test result. As you can easily observe, this part of the input is de facto irrelevant: after the first `TAK` answer, the answers `TAK` must follow until the end of given test case.

Problem F. Fence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Bajtazar is widely believed to be the greatest scrooge in the whole borough. One can find many examples to support this claim, and the least important one of them is that his estate has not even got a fence. However, Bajtazar has recently found n old planks in his basement, so he decided to build at least a fragment of a hoarding.

He stacked the planks one over the other such that their consecutive lengths were $a_1, \ldots, a_n$. He took the first one, cut out a fragment of length b, and nailed it as the leftmost piece of the fence. Then, he cut out the next fragment of length b, and nailed it next to the previous one. He continued doing so until what was left in his hands was a piece of length c ($1 \leq c \leq b$). *Well, this one might seem a little bit too short, but it would be a pity if such a good plank went to waste*, he thought... and added it to the fence as well. He then took the second plank from the stack, then the next one, and repeated the whole procedure for each one of them.

When the job was done, Bajtazar looked at the result... and concluded that using those shorter pieces might really not have been the best idea. *This doesn't look like a cohesive design at all* – he thought – and decided to paint the whole fence white to give it at least a pretence of consistency. *Still*, it occurred to him suddenly, *if I only paint every other plank white, and leave the remaining ones brown, I could use (roughly) twice as little paint and yet the fence will still look like a well thought-out, coherent construction!* And so he only painted every second plank, starting from the leftmost one (apparently, the rumours of Bajtazar's meanness must have been somewhat exaggerated. He could have started from the second leftmost plank after all.).

However, in the evening, a frightening thought struck him: maybe if he had chosen a different value of b, the overall amount of paint used could have been smaller? Well, there's not much that can be done anymore, but just the thought of such an unnecessary wastage keeps Bajtazar awake – he needs to know how much paint he would have used if he had chosen to build a fence of any other possible height. Help him to find the answer so he can finally fall asleep peacefully (or not, depending on the result of your computation).

Input

The first line of input contains the number of test cases z ($1 \leq z \leq 5$).

The descriptions of the test cases follow.

The first line of every test case contains a single integer n ($1 \leq n \leq 1\,000\,000$) – the number of planks. Then follow n integers a_i ($1 \leq a_i \leq 1\,000\,000$, $\sum_{i=1}^{n} a_i \leq 1\,000\,000$) – the lengths of consecutive planks.

Output

For each test case, output M lines, where M is the maximum of all values of a_i within this test case. The i-th line should contain a single integer f_i: the total length of planks which Bajtazar would have needed to paint white if he had chosen the fence height to be $b = i$.

Example

standard input	standard output
1 4 10 7 2 8	14 13 15 13 15 16 21 23 24 12

Note

For height $b = 4$, consecutive fence pickets would have heights:

4 4 2 4 3 2 4 4.

Bajtazar would have needed to paint the total length of $4 + 2 + 3 + 4$, so the answer in the 4-th line is 13.

For height $b = 5$, consecutive fence pickets would have heights:

5 5 5 2 2 5 3.

Bajtazar would have needed to paint the total length of $5 + 5 + 2 + 3$, so the answer in the 5-th line is 15.

Problem G. Gebyte's Grind

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	1024 mebibytes

Like every spring, Gebyte the Witcher embarks on his trail, planning to get a taste of the witcher's craft and earn some coin while doing so. The trail stretches from west to east, n miles long; each mile one of three kinds of objects awaits:

- B b_i: A lair of a cruel beast, a terror known among the local peasants. When Gebyte steps into the lair, the beast within attacks him at once, wounding the witcher. If Gebyte survives the first blow, he quickly draws his silver sword and slains the beast. As a result, Gebyte's health changes according to

  if $H \le b_i$ then *death* else $H := H - b_i$.

- K k_i: A local inn, which Gebyte (known for his weakness for booze) is sure to visit. If he stays at the tavern with health lower than k_i, having drank one barrel too many he dies before dawn. Otherwise, with a heavy hangover, and thus health lowered to k_i, he continues his journey. Gebyte's health changes according to

  if $H < k_i$ then *death* else $H := k_i$.

- C c_i: A house of a powerful witch, whose magic and potions can heal wounds and drive the hangover away. If Gebyte meets the witch while having health below c_i, through witch's spells and decoctions his health raises to c_i. Gebyte's health changes according to

  $H := max(H, c_i)$.

Now, the witcher wonders what part of the trail he should visit, so that he has a bit of fun but keeps his life. Days pass, and on the i-th day one of two things happens:

- One of the objects on the trail changes, for example the witch's house is bought by a local merchant and turned into a tavern, or a new beast crawls out from the underworld, burns an inn down, and sets down a lair in its place;
- Gebyte goes outside his house, sits under a tree, and wonders: if he started his journey at the i-th object, and travelled east, how far would he have gone without losing his life? Such questions are beyond the capabilities of a simple witcher, thus he asks you – a sorcerer known for mysterious coding magic – to answer them.

Note that the witcher only thinks about what to do, without actually embarking anywhere, so while **the changes to the trail stay forever**, **every Gebyte's question is independent** and in each his starting level of health equals H.

Input

The first line of input contains the number of test cases z ($1 \le z \le 100\,000$). The descriptions of the test cases follow.

The first line of a test case contains three integers n, q and H ($1 \le n \le 2\,000\,000$, $1 \le q \le 4\,000\,000$, $1 \le H \le 10^{12}$) – the length of the trail, the number of days, and Gebyte's starting health.

The next n lines describe the initial state of the trail; i-th contains a letter denoting the type of the i-th object (B, K or C), and an integer (b_i, k_i or c_i; $1 \le b_i, k_i, c_i \le 10^{12}$) with the meaning as explained earlier.

The next q lines describe the subsequent days. The i-th line begins with a letter Z if on the i-th day there is a change to the trail, or D if instead Gebyte imagines another journey.

In case of a change to the trail, the rest of the line contains: an integer x_i ($1 \le x_i \le n$), denoting the index of the object undergoing a change, followed by a letter and an integer, in the same format as in the description of the initial state, denoting the new object. In case of Gebyte's imagined journey, the line contains an integer l_i ($1 \le l_i \le n$), denoting the object at which the journey should start.

The total length of all trails and the total number of days do not exceed 2 000 000 and 4 000 000 respectively.

Output

For every test case, output answers for all Gebyte's questions in the order they appear on the input. For every question, output a single integer, denoting the index of the farthest object r_i ($l_i \le r_i \le n$) which Gebyte can get to (and survive), or -1 if he will be killed by the l_i-th object. The answer to the question asked the on i-th day should take into account all changes to the trail from earlier days.

Example

standard input	standard output
1 4 12 10 C 10 B 5 K 5 B 6 Z 3 K 6 Z 1 C 11 D 2 D 1 Z 3 C 1 D 3 Z 3 B 20 D 3 Z 1 C 31 D 1 Z 4 K 6 D 1	2 3 4 -1 3 4

Note

The trail changes six times, in the following manner:

- `[C 10, B 5, K 5, B 6]` (initial state)
- `[C 10, B 5, K 6, B 6]` (1st day)
- `[C 11, B 5, K 6, B 6]` (2nd day)
- `[C 11, B 5, C 1, B 6]` (5th day)
- `[C 11, B 5, B 20, B 6]` (7th day)
- `[C 31, B 5, B 20, B 6]` (9th day)
- `[C 31, B 5, B 20, K 6]` (11th day)

During the remaining six days Gebyte is imagining various journeys.

On the third day, Gebyte starts with the second object. After slaying the beast, he is left with 5 health points – one short to survive the next object (an inn `K 6`). The farthest object Gebyte can reach is thus the second one.

On the fourth day, Gebyte starts with the first object; thanks to the witch he gains one extra health point, and survives the third object (inn), but fails at the last one (beast).

On the sixth day, Gebyte starts with the third object, which is now a witch `C 1`. With his health level unchanged he continues and kills a beast, which is the last object on the trail.

On the eighth day, Gebyte starts with a beast `B 20`, which he cannot defeat, so the answer is `-1`.

On the tenth day, Gebyte start with a powerful `C 31`, thanks to which he defeats the first two beasts, failing at the last one.

On the last day, Gebyte successfully travels through the entire trail.

Problem H. Hidden Password

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

After a successful internship, Bytholomew was hired as a senior cyber-security expert. To lead by example, he decided to finally heed his own advice and use two *different* passwords for his e-mail and for the Facepalm social media. Unfortunately, remembering two passwords proved too much for him. Moreover, he couldn't just plainly write the passwords somewhere, as it would be against another of his recommendations. But being a security expert, Bytholomew knew exactly what to do. He chose his favorite integer $d > 0$ and wrote both passwords encoded with Caesar cipher with key d.

Pleased with his work, he looked at his notes and the horrible truth dawned on him: after the encoding, the first (e-mail) password became literally the second (Facepalm) password, while the second one turned into the first one. „Holy moly!" – Bytholomew exclaimed, as there was nothing more to say.

Now you too can become a security expert – knowing the first of Bytholomew's passwords, guess the second one, if possible.

Input

The first line of input contains the number of test cases z ($1 \le z \le 20$). The descriptions of the test cases follow.

Each test case is one word – the password – consisting of lowercase English letters, in a separate line. The password has at least 1 character and at most 200 000 characters.

The total number of characters in all passwords does not exceed 1 000 000.

Output

For every given password, guess and output the second one in a separate line. If the second password cannot be determined (either because there is no solution or because there is more than one), output a single word `NIE` instead.

Example

standard input	standard output
1 cnffjbeq	password

Note

The Caesar cipher means substituting each letter with the one d places down the alphabet, treating alphabet as cyclic if necessary. E.g. for $d = 3$ the letter `a` is substituted by `d`, `b` by `e`,...., `w` changes into `z`, `x` to `a`, `y` to `b` and `z` to `c`.

Problem I. Interesting Numbers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

The *bitwise exclusive or*, or simply XOR, is an operation denoted by $\oplus$ which works on two integers by XOR-ing their corresponding bits: if x_i, y_i, z_i denote the i-th binary digit of x, y and z, where $z = x \oplus y$, then $z_i = (x_i + y_i) \mod 2$.

You are given a positive integer k. A sequence of integers is *interesting* if XOR of any its two elements is less than or equal k.

Given a sequence $a_1, \ldots, a_n$, determine the largest possible length of its interesting subsequence. (A subsequence is a sequence that can be derived from the given sequence by deleting zero or more elements.)

Input

The first line of input contains the number of test cases z ($1 \le z \le 1\,000$). The descriptions of the test cases follow.

The first line of each case contains two integers n, k ($1 \le n \le 30\,000$, $1 \le k < 2^{20}$) – the length of the sequence, and the upper bound on maximum XOR of its two elements.

The second line contains n nonnegative integers $a_1, \ldots a_n$ ($0 \le a_i < 2^{20}$) – the given sequence described above.

The sum of n and k over all test cases do not exceed 200 000 and 3 200 000 respectively.

Output

For every test case output a single integer – the maximum possible length of an interesting subsequence of the given sequence.

Example

standard input	standard output
1 7 11 3 12 9 10 16 3 4	4

Note

The elements 3, 9, 10 and 3 form an interesting subsequence, as XOR of every pair is not larger then 11. For example $9 \oplus 10 = 1001_2 \oplus 1010_2 = 11_2 = 3 \le 11$. There is no subsequence consisting of five elements with the same property, e.g. the sequence $(3, 9, 10, 3, 4)$ is not interesting because of $4 \oplus 9 = 100_2 \oplus 1001_2 = 1101_2 = 13 > 11$.

Problem J. Jungle Trail

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

In the mobile game "Jungle Trail", you are given a rectangular $n \times m$ board divided into $n \cdot m$ squares. Each square is either empty, blocked (impassable) or contains a den of snakes, either poisonous or benign (not poisonous). If a square contains a den of snakes, then either all the snakes on a given field are poisonous, or all are benign.

The game allows you to tap any column or any row of the board. If you tap a column, all poisonous snakes in this column are turned to benign, and vice versa. Similarly, if you tap any row, all snakes in the row change their state. You can tap each row/column only once. If a den is in a tapped row as well as in a tapped column, its state returns to the original one.

After performing all those operations, you must find a trail through the jungle: a path which starts at the top left corner, in every move goes either one square down or one to the right, ends at the bottom right corner and never passes through a den of poisonous snakes or a blocked field.

Input

The first line of input contains the number of test cases z ($1 \le z \le 500$). The descriptions of the test cases follow.

The first line contains two integers n and m ($2 \le n, m \le 2\,000$).

Each of the following n lines contains m characters `.`, `#`, `O` (capital o) and `@` (at sign), meaning an empty square, blocked square, den of benign snakes and den of poisonous snakes, respectively. You may assume that the top left corner and the bottom right corner are not blocked.

Neither the sum of n values over all test cases nor the sum of all m values exceed 15 000.

Output

For every test case output the solution in the following format:

The first line should contain `TAK` if a jungle trail is possible or `NIE` if it isn't.

If the answer is `TAK`, in the next three lines output:

- A sequence of n characters T or N, the i-th character being T if the i-th row should be tapped, N if not; - A sequence of m characters T or N, determining in the same way whether the columns should be tapped; - A sequence of $n+m-2$ characters P or D denoting the trail: P means a move right, D means a move down.

Example

standard input	standard output
1 4 5 ..#.. @@O@@ ##@#O ..@.@	TAK NTNN NNTNT DPPDDPP

Note

After tapping the rows and columns described on the output, the board is in the following state:

```
..#..
```

```
OOOO@
##O#@
..O.O
```

Now the given path goes only through `.` and `O` squares.

Problem K. Kitten and Roomba

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15 seconds
Memory limit:	1024 mebibytes

A little kitten of Bituś, Kapitan, does love sleeping! Sadly, the average quality of Kapitan's sleep has decreased significantly after its owner decided to buy Roomba – a robot vacuum cleaner. It seems the kitten is scared of Roomba as... well, it is fairly scared.

Bituś's house consists of n rooms connected by $n-1$ two-way corridors in such a way that it is possible to reach any room from any other one. Bituś noticed that whenever Roomba enters a room with Kapitan inside, the kitten awakes instantly and runs away to one of the neighbouring rooms, where it returns back to dreaming about playing with mice. Frightened Kapitan escapes the room blindly, so if there exists more than one room directly connected to the current one, then *every neighbouring room is equally likely to chosen by Kapitan* (in particular, it can escape to the room from which Roomba has just came from).

During one particularly long night shift at work, Bituś opened the Roomba app and observed that during today's cleaning it visited rooms $a_1, \ldots, a_m$ (in this order). A room can appear more than once in this sequence, but every two neighbouring rooms must be directly connected to each other. Bituś also remembers that initially the kitten had been sleeping in the room c. Moreover, it must hold that $a_1 \neq c$ because observant Kapitan would never ever sleep in one room with Roomba!

Now Bituś wonders what is the *expected value* of the number of times Roomba has woken Kapitan during the cleaning session. Please help Bituś to find the answer so he can finally return back to work.

Input

The first line of input contains the number of test cases z ($1 \leq z \leq 6\,000$). The descriptions of the test cases follow.

The first line of a test case contains two integers n, c ($2 \leq n \leq 1\,000\,000$, $1 \leq c \leq n$), the number of rooms in Bituś's house and the identifier of the room where Kapitan sleeps initially.

The following $n-1$ describe corridors. Each of them contains two integers u_i, v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$) signifying that the rooms u_i and v_i are connected. You can assume that you can reach every room from any other.

The next line contains the number of rooms m ($1 \leq m \leq 5\,000\,000$) visited by Roomba during the cleaning session.

The last line of the test case contains a sequence of m integers a_i ($1 \leq a_i \leq n$) – the rooms visited (in this order) by Roomba. Every two subsequent rooms are connected with a corridor; moreover, assume that $a_1 \neq c$.

The sum of values of $n+m$ over all test cases does not exceed $12\,000\,000$.

Output

For every test case print one real number e – the expected number of times Roomba entered a room with Kapitan inside. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-5}. Namely, if your answer is a, and the correct value is b, then your answer will be accepted if $\frac{|a-b|}{\max(1,b)} \leq 10^{-5}$.

Example

standard input	standard output
1 4 2 1 2 2 3 4 2 4 1 2 3 2	1.666666666666667

Problem L. Lemurs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

Bajtazar has been sent on a mission to the jungle. He is going to study the behaviour of newly discovered species – taxicab lemurs.

The jungle has a rectangular shape with dimensions n by m. Taxicab lemurs are already known to be group animals. Furthermore, each group of lemurs inhabits only one cell of the jungle, i.e. a single cell with coordinates (x, y), where $1 \leq x \leq n$ and $1 \leq y \leq m$. But the most fascinating aspect of their behaviour lies in their foraging area: if some group of lemurs inhabits a cell with coordinates (x, y), then their foraging area is bounded by a ball with radius k in the taxicab metric. In other words, the foraging area of this group of lemurs is the set of cells with coordinates (x', y') satisfying $|x - x'| + |y - y'| \leq k$ and $1 \leq x' \leq n$, $1 \leq y' \leq m$. The constant k is universal for all taxicab lemurs.

The last taxicab lemur's researcher has disappeared under unknown circumstances and the only thing he left behind was an n by m map with some cells marked as taxicab lemurs' foraging area (the circumstances of this event are still unclear; the latest theory suggests that the newly discovered species of lemurs may not be herbivorous after all...). Slightly disturbed by that story, Bajtazar is currently wondering if this map is reliable. Therefore he would like to check if there exists such a set of groups of taxicab lemurs, that their foraging area would match precisely the cells marked on the map.

Help Bajtazar and find the answer to his bugging question. He will certainly be grateful till his very last days!

Input

The first line of input contains the number of test cases z ($1 \leq z \leq 4\,000$). The descriptions of the test cases follow.

The first line of each test case consists of numbers n, m, k ($1 \leq n, m, k \leq 1\,000$) with meaning as described in the task statement.

The next n lines describe the map. If, according to the map, the taxicab lemurs forage on a cell with coordinates (j, i), then in the i-th line on j-th position will be a character `x`, otherwise this character will be a dot.

The sum $n + m + k$ over all the test cases does not exceed $100\,000$.

Output

For each test case print a single line. If the answer to Bajtazar's question is yes, print `TAK`, otherwise print `NIE`.

Example

standard input	standard output
2 3 3 1 .xx xxx xx. 3 4 1 ..xx x.xx x..x	TAK NIE

Note

In the first test case, the marked foraging area is generated by a set of 3 groups of lemurs inhabiting cells with coordinates $(1, 3)$, $(2, 2)$ and $(3, 1)$.

In the second test case, there does not exists such a set of groups of lemurs that would generate the foraging area described by the map.

Problem M. Median

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

Yaawn... you really shouldn't have played on ByteStation till 3 AM. You begin to slowly drift away, immersed by the monotone voice of your algorithms professor...

Wait. What happened? It seems that you fell asleep during online classes, and slept through the entire afternoon, night, and the next morning. But that means... the exam! Barely catching your breath, you bust into the classroom precisely when the professor is beginning to explain the problem statement.

The task at the exam is to compute the median of a sequence, defined as the middle element in the sorted order (if the length of the sequence is even, the median is the smaller out of the two middle values). You're supposed to write down the pseudocode of a solution, and then simulate it on an example sequence provided by the professor.

Reaching into the depths of your memory, you seem to recall something like that appearing during the lecture. Some sort of a magic algorithm... magic threes? Yaawn, the memory itself makes you sleepy again. You somehow split the sequence into parts, solve them recursively, and then combine...?

Based on the bits and pieces that you remembered, you came up with the following algorithm:

```
function magicThrees(sequence)
  if the length of the sequence is no more than 2 then
    return the smallest value in sequence
  otherwise
    part_1, part_2, part_3 = splitIntoThreeParts(sequence)
    median_i = magicThrees(part_i) dla i = 1, 2, 3
    return the median of [median_1, median_2, median_3]
```

where `splitIntoThreeParts` divides the sequence into three connected fragments, with lengths as close to each other as possible. Concretely, the fragments will have lengths `[s, s, s]`, `[s + 1, s, s]` or `[s + 1, s + 1, s]`, depending on the length of the original sequence. For example, the sequence `[8, 2, 6, 6, 3, 5, 7, 1]` will be divided into `[8, 2, 6]`, `[6, 3, 5]` and `[7, 1]`.

After leaving the exam, you realized that your algorithm is not so magical after all, as it doesn't always work. **Maybe, at least, it has worked on the example sequence from the exam**... Unfortunately, your memory of that sequence is as fuzzy as the one of the algorithm itself: while you do remember **almost all** elements of the sequence from the exam, you're not sure about a few of the values. However, you do remember the overall bounds on all values appearing in the sequence: all elements were supposed to lie in a (closed) interval $[0, m-1]$.

Calculate the number of ways to choose the values you don't know, in such a way that `magicThrees` executed on the resulting sequence returns the correct median (as defined above). As the answer may be very large, it's enough if you find its remainder modulo $10^9 + 7$.

Input

The first line of input contains the number of test cases z. The descriptions of the test cases follow.

The first line of a test case contains two integers n, m ($n \geq 1$, $1 \leq m \leq 10^9$), denoting the length of the test sequence and the bound on the values of its elements.

The second line contains the test sequence, described as n integers from the range $[-1, m-1]$, where elements with unknown values are denoted by -1.

If we denote the number of unknown values by q, then every test case belongs to one of the following

groups:

- $1 \le z \le 100$, $1 \le q \le 10$, $n \le 3^4 = 81$
- $z = 15$, $1 \le q \le 20$, $n \le 3^5 = 243$
- $z = 3$, $q = 30$, $n \le 3^8 = 6\,561$

Output

For every test case, output a single integer r ($0 \le r < 10^9 + 7$) – the answer to the question posed in the problem statement.

Example

standard input	standard output
3 3 10 -1 -1 3 4 50 10 20 -1 40 5 100 -1 10 10 -1 20	100 21 1979

Note

In the first test case, `magicThrees` returns the correct median irrespective of the two unknown values; the answer is thus $10^2 = 100$.

In the second test case, `magicThrees` returns the correct median if and only if the unknown value is not larger that 20, which gives 21 ways.

In the third test case, `magicThrees` returns an **incorrect** median if both unknown values are smaller than 10, or both larger, which gives $100^2 - (10^2 + 89^2) = 1979$ ways.