Algolympics 2023 Online Elimination Round

A. The Encrusted Lock

1 second, 256 megabytes

The commander of the Imperial Fleet of Vega is visiting the Star System Sol on an inquisition. "So, who should I blast to the void?", asks the commander. "I'm sure it's one of you dungeon keepers."

The dungeon within the Star System Sol is known to host the most notorious of criminals in the Milky Way Galaxy. Recently, a high-profile criminal has fled, managing to escape from his chambers and hijack a cruiser ship. As the entire facility was heavily guarded, the conclusion was that it was an inside job.

You are his Lieutenant, Pladius Lapiz, and it is your job to figure out which of these dungeon keepers are guilty. "It would be so much easier if any of you clean around here", you mutter as you investigate the escapee's chambers. You take a look at the chamber's primitive lock, a remnant of the old inhabitants of Sol.

"Nothing on one, two is binding, click on three".

You notice that the lock has been encrusted with a lot of dirt, so there is a bit of leeway with which keys are capable of opening it.

As someone familiar with locks, you immediately take note of the length of the keyhole D, meaning the key should have length D. You also note the depth of groove i in the keyhole as A_i. Lastly, you measure the depth of the dirt buildup in each of the groove as B_i.

From your experience, you determine that some key can open the lock if and only if the following is true for each i from 1 to D: the height of its ith tooth is at least $A_i - B_i$ and at most A_i (this happens because the dirt adds to the teeth of the keys).

There are N dungeon keepers, each with one key. Each key is defined by D integers, denoting the heights of each of its teeth, in order. With your findings, you figure that you might narrow down the suspects. A dungeon keeper is still suspicious if their key can open the lock; otherwise, they are surely innocent.

Which dungeon keepers are still suspicious?

Input

The input will only contain one test case.

The first line contains two space separated integers, D, denoting the depth of the keyhole, and N, the number of dungeon keepers.

The second line contains D space separated integers $A_1, A_2, \ldots, A_D$, denoting the depth of each groove in the keyhole.

The third line contains D space separated integers $B_1, B_2, \ldots, B_D$, denoting the depths of the dirt in each groove in the keyhole.

Then, N lines then follow, with each one describing the key of each of the groundskeepers. The line consists of D integers, where the ith integer in each line corresponds to the height of the ith tooth in the key of that groundskeeper.

Constraints

- $1 \leq N \leq 10$
- $1 \leq D \leq 10^4$
- $0 \leq B_i \leq A_i \leq 10^6$
- $0 \leq \text{heights of each tooth} \leq 10^6$

Output

Output N lines. The ith line should contain `SUS` if the ith groundskeeper should remain a suspect for the theft, or the word `INNOCENT` otherwise.

input

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

output

```
SUS
SUS
INNOCENT
SUS
INNOCENT
```

The second dungeon keeper has a key with teeth $1, 4, 5, 2, 1$, which fits the criteria of each tooth being in the bounds of $[1, 1]$, $[3, 5]$, $[4, 7]$, $[1, 2]$, and $[0, 3]$, respectively.

In contrast, the third dungeon keeper has a key with the first teeth 5 not within the bounds, therefore they are innocent.

SAMSUNG Research

This problem is contributed by Samsung R&D Institute Philippines

B. Ray Ricochet

1 second, 256 megabytes

(Please note that this is a work of fiction, and does not reflect actual stellar dynamics)

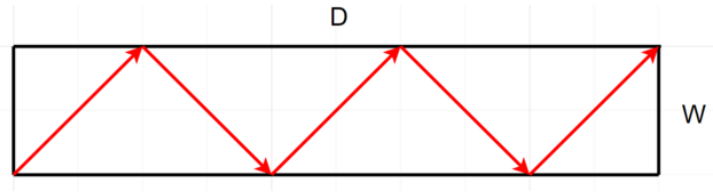
Due to the pull of gravity between two nearby stars, a phenomenon known as a "space corridor" is formed in the space between them. The space corridor is modeled as a rectangle, with "walls" induced by gravity. The width of this space corridor is the distance between the stars W, and the depth of this corridor is some value D. Please refer to the diagram so that it's absolutely clear which dimension is measured by D, and which one is measured by W.

If left alone, this corridor will collapse, eventually colliding in a stellar fashion (get it?). The Intergalactic Star Preservation Department is tasked with preventing such things from happening.

To do just that, they have invented a repulsion ray that will be fired deep into the corridor. They ray can bounce along the walls of the space corridor. The process starts with firing the repulsion ray at a 45 degree angle from one of the corners of the space corridor, towards the other end of the corridor. When the ray hits a wall of the space corridor, it reflects off it such that the angle of incidence is equal to the angle of reflection.

The ray can bounce along the corridor walls up to B times before it dissipates, and it must reach the other end of the corridor before that. This value B varies from corridor to corridor. Note that reaching the end of the space corridor, even at a corner, does not count as a bounce.

The ray can bounce along the corridor walls up to B times before it dissipates, and it must reach the other end of the corridor before that. Note that reaching the end of the space corridor, even at a corner, does not count as a bounce, as the ray is already able to do its job.



Drawn diagram of a space corridor with a repulsion ray bouncing inside it. $D = 5$ and $W = 1$. We count 4 bounces.

Given a space corridor's depth D, its width W and the maximum number of ray bounces B, determine whether the space corridor collapse can be prevented.

Also, you will answer multiple independent test cases, and the maximum number of bounces B can vary from case to case.

Input

The first line contains a single integer T denoting the number of test cases.

Each test case consists of a single line with 3 space separated integers, D, W, and B, the depth of the space corridor, the width of the space corridor, and the maximum number of bounces.

Constraints

- $1 \leq T \leq 100$
- $1 \leq D, W, B \leq 10^5$

Output

For each test case, output a single line containing `YES` if the space corridor can be prevented from collapsing, or `NO` if not.

input

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

output

```
YES
NO
YES
```

For the first test case, it will reach the end of the space corridor after two bounces.

The third case is shown in the figure above. The ray can reach the end in 10 or fewer bounces.

SAMSUNG Research

This problem is contributed by Samsung R&D Institute Philippines

C. Travelling Through Space-Time

2 seconds, 256 megabytes

The war against the infamous terrorist organization, NELOC, has left the galaxy in shambles. Of the K star systems that fought in this war to save humankind, several have already collapsed, and the remaining forces are at their wits' end. As a last ditch effort, humankind has developed space-time travel, which enables them to send a short message to a specific star and backwards in time.

The K star systems are arranged in a row. Sending a message to a star that is doomed for destruction can prevent its collapse. However, sending a message to a supposedly surviving star will do the opposite and drive them towards their collapse. In other words, by sending a message to a star, you flip its state from surviving to collapse and vice versa.

Moreover, sending a message to a star triggers a "butterfly effect" and also flips the state of all other stars after it.

For example, suppose there are 8 star systems and their states are the following, in order, where `1` denotes a surviving star, while `0` denotes a collapsing one:

```
11110001
```

Then sending a message to the 5th star will result in the following states:

```
11111110
```

The state of the 5th star is reversed, but so are the states of all the stars that come after it.

Your only hope to win against NELOC is to ensure that all star systems survive. To do that, how many messages do you need to send back in time? The smartest minds in the galaxy have assured you that this task is always possible.

Input

Input will begin with an integer T denoting the number of test cases. T test cases follow.

Each test case is made up of two lines.

The first line contains a single integer K denoting the number of star systems.

The second line in each test case contains a single string made up of K digits, each either `0` or `1`, describing the states of the star systems, in order. If the ith character is `1`, then that means that the ith star system survived, while a `0` means it has collapsed.

Constraints

- $1 \leq T \leq 1000$
- $1 \leq K \leq 100$

Output

For each test case, output a line containing a single integer, the minimum number of times a message must be sent back to the past in order to save all star systems (in that test case).

input
3 8 11110001 3 100 8 00111001
output
2 1 4

For the first sample case, we can send a message to the 5th star system, then the 8th. This will result in the following state transitions:

```
11110001
11111110
11111111
```

SAMSUNG Research

This problem is contributed by Samsung R&D Institute Philippines

D. Graffiti Math

5 seconds, 256 megabytes

Detective Blanc passes by a nearby star. "Another one here", he mutters as he takes note of graffiti somehow embedded onto the surface of the star.

The graffiti is a mystery – no one knows who puts them there or how they embed them onto the surface of the stars. But Detective Blanc suspects that the series of star graffiti form a code that could lead him to a notorious group of NELOC smugglers.

From your observations, you note that all of the graffiti contain characters in different stars of the galaxy. These characters are arranged in a similar fashion to math equations with two operands. However, instead of numbers, they use capital letters instead. An example equation in one of the graffitis is located below.

```
U+ME=LOV
```

You suspect that this is a code that will allow you to decipher other text present on the graffiti. Your intuition and experience with ciphers tell you that each letter in the equation represents a digit from `0` to `9`. When the letters are replaced with their corresponding digits, and the resulting strings interpreted as numbers, the equation must also be satisfied.

Suddenly, a message arrives from an informant inside NELOC with extra information.

1. For the subtraction operation, the first operand is always larger than the second.
2. For the division operation, use floor division (ignore remainder).
3. Leading zeroes are allowed (possibly multiple).
4. For every digit, there is at most one corresponding letter. And similarly for every letter, there is at most one corresponding digit.
5. All operands and results are **positive** (non-zero) integers when translated to their numerical form.
6. In case multiple possible character to numeral assignments are applicable, select the one that assigns the lowest digit value to the character that appears earliest in the alphabet, and so on.
7. There is always a solution. Find it.

With this information, determine the character to numeral assignments that satisfy the graffiti equation.

Input

Input will begin with an integer T denoting the number of test cases. T test cases follow.

Each test case consists only of two lines.

The first line contains a single integer C, denoting the number of unique characters in the equation.

The second line contains the graffiti equation in the format below

```
<operandA><operation><operandB>=<result>
```

The operation here is one of `+`, `-`, `*`,`/` corresponding to addition, subtraction, multiplication, and division respectively. The operands and results are strings of uppercase letters.

Constraints

- $1 \leq T \leq 10$
- $1 \leq C \leq 9$
- $1 \leq \text{length}(\text{operand}, \text{result}) \leq 10$

Output

For each test case, output C lines, with each line containing a character and its corresponding numeral separated by a space. Output the characters in alphabetical order.

input
2 5 ADE-BC=CC 6 U+ME=LOV

output

```
A 0
B 1
C 2
D 3
E 4
E 2
L 0
M 3
O 4
U 9
V 1
```

The equations $034 - 12 = 22$ and $9 + 32 = 041$ are both valid (despite the leading zeroes).

SAMSUNG Research

This problem is contributed by Samsung R&D Institute Philippines

E. Modular Zeroes

2 seconds, 256 megabytes

Due to a recent prison escape, the dungeon of the Star System Sol has upped their security. They are now making use of an AI system to lock their prison chambers.

"`How many zeroes?`" *the AI system, Gustov Macintlog, asks Inspector Mont. Mont Blanc just scratches his head and replies, "Not sure what you mean, Monsieur Gustov; this state-of-the-art security system is very complex!"*

The security system for the dungeon chambers has N combination locks, each with M symbols inscribed on them. These locks may seem extremely important at first but they are merely red herrings! The true way to get into the vault is by talking to the AI system, Gustov Macintlog, and giving the correct answer to his question.

You jump back to the middle of your conversation with Macintlog. "To put it simply", Mont adds, "the device has many locks on it, added one by one. It starts with a complexity of 1 after adding the 1st lock, and after that, the k th added lock causes the complexity of the system to increase by k times! So if you have three locks, the resulting complexity will be 6, since $1 \times 2 \times 3 = 6$"

AI Gustov: "`How many zeroes...at the end?`"

Mont Blanc: "... In the example I just gave you, none. The complexity of 6 does not have any zeroes at the end"

AI Gustov: "`Incorrect`"

Mont Blanc: "Wait, I remember something from reading your manual. You speak in base-M right? In that case, since there are $M = 3$ symbols, doesn't 6 become $20_{\text{base }3}$?

AI Gustov: "`Correct`"

Mont Blanc: "Then that has 1 trailing zero, gotcha!"

AI Gustov: "`Correct`"

The prison chamber finally opens.

Given N and M, the complexity C of the system would be $1 \times 2 \times \cdots \times N$. If C is represented as a base-M number, how many *trailing zeroes* would there be?

Of course, *your* answers should always be in base ten! The input values are all in base ten as well, don't worry.

Input

Input will begin with an integer T denoting the number of test cases. T test cases follow, each in one line.

Each test case consists of two space-separated integers N and M (both represented in base ten).

Constraints

- $1 \leq T \leq 10^4$
- $1 \leq N \leq 10^{15}$
- $2 \leq M \leq 10^9$

Output

For each test case, output a line containing a single integer in base ten, denoting the number of trailing zeroes in the base-M representation of the complexity C of the system containing N locks.

input

```
4
3 3
10 2
69 12
734206856189756 848279527
```

output

```
1
8
32
865524
```

SAMSUNG Research

This problem is contributed by Samsung R&D Institute Philippines

F. Star Alchemy

2 seconds, 256 megabytes

Patrick Star is a cosmic entity from a higher plane, trying to take over the universe through the fusion of stars, the process known as Star Alchemy.

For his plans of galactic conquest, he will need to produce Q stars of possibly different types.

Star Alchemy is an expensive process. Each star can be fused from scratch, but fusing a star from scratch costs a lot of energy, in the scale of Terajoules. Good thing that some stars have another way that they can be fused – which is to follow some recipe in order to fuse several other stars of possibly different types into one. You still need to procure the constituent stars in turn, however. Each star has at most one such recipe.

For example, let's say a Red star costs 1000TJ of energy to fuse from scratch. If the same Red star can be made using 3 Green stars (each costing 50TJ to fuse from scratch) and 2 Blue stars (each costing 100TJ from scratch), then it is more economical to create the Red star from Green and Blue stars. Note that this fusion can go even further down – if Green/Blue stars can also be fused for cheaper, then it will also affect the total cost of the final Red star.

Given this, can you help Patrick find the cheapest cost to fuse (either from scratch or by combination) each of the required stars he needs to take over the universe?

Also, Patrick is a bit of a klutz, and his recipe notes are all scattered! You're going to need to reconstruct those, too.

Input

There is one test case per file.

The test file begins with three positive integers N (number of different stars), R (number of requirements for recipes), and Q (needed stars).

Then, N lines follow, each containing a star description. The ith star description is composed of an alphanumeric string S_i and an integer C_i, which are the name of the ith star type and the energy (in TJ) required to fuse it from scratch, respectively.

Then, R lines follow, each containing a requirement for a recipe, as described below.

The kth requirement is represented as two alphanumeric strings Res_k, Req_k and an integer M_k, all space-separated.

1. Res_k denotes the resulting star
2. Req_k denotes the requirement star
3. M_k denotes how many of the requirement star is needed to create the resulting star.

Or, in other words, "Fusing Res_k out of smaller stars requires exactly M_k copies of Req_k.
For example, `red green 3` means that we need 3 green stars to create a red star.

Recall that each star has at most one recipe for fusing it, so if the same resulting star appears across multiple requirements, this means that **all** listed requirement stars are needed to create one such resulting star (see the explanation of the sample I/O for an example).

Likewise, it is also possible that a star has no such recipe and can only be fused from scratch, which is the case if some star never appears as a resulting star in this input.

Finally, Q lines follow, describing the stars that Patrick needs. The ith such query consists of an alphanumeric string q_i, the name of the star that Patrick needs.

Constraints

- $1 \leq N, Q \leq 10^4$
- $1 \leq R \leq 10^5$
- $1 \leq C_i, M_k \leq 10^9$
- The length of each S_i, Res_k, Req_k, q_i is at least 1 and at most 5, and they are all composed of alphanumeric characters.
- Res_k, Req_k, q_i are valid star types (an S_i exists that will match them)
- Res_k will never be the same as Req_k per requirement (i.e. it will never require itself)
- Crafting a star will never require a star of the same type, whether directly or indirectly.

Due to the test data being possibly huge, fast I/O is recommended for this problem.

Output

Output Q lines, answering each of Patrick's questions. In the ith line, output the cheapest cost to fuse the star q_i by any means necessary.

input

```
3 2 1
red 1000
blue 100
green 50
red blue 2
red green 3
red
```

output

```
350
```

Since both `red blue 2` and `red green 3` appeared as requirements, fusing a single red star from other stars requires exactly 2 blue stars **and** exactly 3 green stars. We see that fusing a red star out of blue and green stars is cheaper than fusing one from scratch (the blue and green stars were fused from scratch, though).

SAMSUNG Research

This problem is contributed by Samsung R&D Institute Philippines

G. Redundant Metronomes

1.5 seconds, 256 megabytes

Did you know that pulsars are neutron stars that emit electromagnetic radiation at extremely regular intervals? They're so regular that the Cosmic Forces of the Universe have decided to use them as metronomes!

The One Above All has a collection of n pulsar metronomes, synchronized such that all of them pulsed together exactly at time 0. Each metronome then proceeds to pulse regularly according to its tempo. The ith metronome has a tempo value of b_i, meaning it then emits a pulse of radiation once every b_i milliseconds. For example, a metronome whose tempo value is 3 will emit pulses at times $0, 3, 6, 9, 12, 15, 18, \ldots$ milliseconds, and so on, whereas a metronome whose tempo value is 6 emit pulses at times $0, 6, 12, 18, 24, 30, 36, \ldots$ milliseconds, and so on.

The One Above All then realized something. If he wanted a beat every 6 milliseconds, he could use a metronome with tempo value 6, *or* he could just as well use a metronome with tempo value 3 then ignore every other click. So, if The One Above All has a metronome with tempo value 3 and a metronome with tempo value 6, he says that the former makes the latter redundant. Generally speaking, metronome A makes metronome B redundant if the metronome A also clicks whenever metronome B does.

The redundancy value of a metronome in The One Above All's collection is equal to the number of *other* metronomes in his collection that make it redundant (a lower redundancy value means this metronome is more near irreplaceable). Given the tempo values $b_1, b_2, b_3, \ldots, b_n$, find the redundancy value of every metronome in the collection.

Input

Input begins with a single line containing the integer n.

The next line of input contains the n space-separated integers b_1, b_2, b_3, ..., b_n.

Constraints

- $2 \leq n \leq 5 \times 10^5$
- $1 \leq b_i \leq 5 \times 10^6$

Due to n being possibly huge, fast I/O is recommended for this problem.

Output

Output a single line containing n space-separated integers, where the ith of these is the redundancy value of pulsar metronome i (the one with tempo value b_i).

input
7 3 14 1 5 9 2 6
output
1 2 0 1 2 1 3

input
6 2 2 3 3 3 6
output
1 1 2 2 2 5

H. I Cast Arcane Missiles

2 seconds, 256 megabytes

While the more affluent space travellers can afford to use their fancy technologies like hyperspace drives and wormhole abuse, us working class space travellers have to do interstellar travel the long way. And by "the long way", I mean "several thousand light years of waiting". Joyous.

To pass the time, you've decided to pick up a new mobile game. It's called Heart of Stone, and it's got a lot of crazy randomness in it.

You have n enemies. The ith enemy has an attack value a_i and a health value h_i. Now, you cast the spell *Arcana Force: Missiles*, which fires m missiles that randomly target the enemies. Formally, we repeat the following step m times:

- Uniformly randomly select one of the still-living enemies and decrease its health value by 1.
- If this leaves it with 0 health, the enemy dies (and can no longer be selected by the random process in future iterations).
    - If *all* enemies are dead, then no more missiles are fired and the process immediately ends.

You must answer T test cases. For each one, you are given n and m, the attack and health of each of the n enemies, and some value x. Find the **exact** probability that after firing m missiles, the enemies that are still alive have a total attack that is $\leq x$.

Input

The first line contains a single integer T denoting the number of test cases.

Each test case begins with a single line containing three space-separated integers n, m and x.

This is followed by a line of n space-separated integers a_1, a_2, ..., a_n, the attack values.

This is followed by a line of n space-separated integers h_1, h_2, ..., h_n, the health values.

Constraints

- $1 \leq T \leq 3000$
- $0 \leq x \leq 10^9$
- $1 \leq n, m \leq 7$
- $1 \leq a_i \leq 10^8$ for all i
- $1 \leq h_i \leq 10$ for all i

Output

For each test case, output a line containing the answer as an **exact** probability.

Suppose that the probability is p/q, where p and q are integers and the fraction is in lowest terms (note that $0 = 0/1$ and $1 = 1/1$). Output p, then a forward slash $/$, then q.

input
3 2 7 5 6 6 1 1 4 3 8 9 9 9 9 1 1 1 1 3 2 8 9 4 4 1 1 2
output
1/1 0/1 11/18

Let's examine the third test case. Suppose the enemies' health values are $\{1, 1, 2\}$, we're firing 2 missiles, and we want the probability that the first enemy dies.

- There is a $1/3$ chance that the first missile hits enemy 1, killing it.
- There is a $1/3$ chance that the first missile hits enemy 2, killing it. Then, there is a $1/2$ chance that the second missile hits enemy 1, killing it.
- There is a $1/3$ chance that the first missile hits enemy 3, which does *not* kill it. So, there is a $1/3$ chance that the second missile hits enemy 1, killing it.

Together, there is a $1/3 + 1/6 + 1/9 = 11/18$ chance that the first enemy is killed by the missiles.

I. Star Travel α

2 seconds, 256 megabytes

Uh oh! The Infinite Improbability Drive on the Heart of Gold spaceship has been severely damaged ever since you drunkenly attempted to use it to try to find an answer to P=NP.

In order to repair it, we're going to need to gather materials from n different stars that are labeled 1, 2, 3, ..., n. Recall that the Infinite Improbability Drive allows you to instantaneously transport yourself to any point in the universe in a mere nothingth of a second, so it would normally take no effort at all to just visit each star.

However, the IID is a bit faulty due to being damaged—the crew on the Heart of Gold have identified m different *paradox conditions*. The ith of these paradox conditions is described by two integers u_i and v_i as follows:

- If you visit star v_i but **haven't** visited star u_i yet, then we enter a probability-zero timeline and the universe collapses.

Your goal is to find a permutation $p_1, p_2, p_3, \ldots, p_n$ of the integers from 1 to n, such that we can visit the stars in that order (visiting star p_1 first and star p_n last) without causing the universe to collapse.

Now, supposing it is possible, there might still be many different valid orderings. Among them, you should pick the most *natural one*. The crew of the Heart of Gold have determined when one ordering is more natural than another:

- Suppose A and B are valid permutations. Let i be the *first* place where these permutations differ, i.e. let i be the *smallest* index (from 1 to n) such that $A_i \neq B_i$. Then, A is more natural than B if and only if $A_i > B_i$.

You can check the explanation of the sample I/O for some concrete examples.
Among all possible valid permutations that dictate in which order to visit the stars (without causing the universe to collapse), output the most natural one. Or, report the task is impossible if no such valid permutation exists.

Input

The first line contains a single integer T denoting the number of test cases.

Each test case begins with a single line containing two space-separated integers n and m, the number of stars and the number of paradox conditions. This is followed by m lines that each describe a paradox condition: the ith of these lines contains the two space-separated integers u_i and v_i.

Constraints

- $1 \leq T \leq 1000$
- $2 \leq n$ and $0 \leq m$
- $2 \leq \sum n \leq 2 \times 10^5$ and $0 \leq m \leq 2 \times 10^5$ across all test cases
- For all paradox conditions (for all i), $u_i \neq v_i$.
- For all pairs of paradox conditions within the same test case (for all i, j), if $i \neq j$, then $u_i \neq u_j$ or $v_i \neq v_j$.

Output

For each test case, output a line with n space-separated integers, the most natural permutation that is valid; or output `IMPOSSIBLE` if no such valid permutation exists.

input
2 5 4 2 1 2 4 4 5 1 5 2 2 1 2 2 1
output
3 2 4 1 5 IMPOSSIBLE

For the first test case, there are 10 possible permutations that do not cause the universe to collapse. We list them here, from most to least natural.

- $\{3, 2, 4, 1, 5\}$
- $\{3, 2, 1, 4, 5\}$
- $\{2, 4, 3, 1, 5\}$
- $\{2, 4, 1, 5, 3\}$
- $\{2, 4, 1, 3, 5\}$
- $\{2, 3, 4, 1, 5\}$
- $\{2, 3, 1, 4, 5\}$
- $\{2, 1, 4, 5, 3\}$
- $\{2, 1, 4, 3, 5\}$
- $\{2, 1, 3, 4, 5\}$

For example, we see that $\{3, 2, 4, 1, 5\}$ is more natural than $\{3, 2, 1, 4, 5\}$ because they first differ at the third visited star, and $4 > 1$. It is also more natural than $\{2, 4, 3, 1, 5\}$ because they already differ at the first visited star, and $3 > 2$.

J. Star Travel β

2 seconds, 512 megabytes

The Grebulons are at it again! Before they enter long-term stasis, they have decided to make it their goal to visit some of the most significant stars throughout human history.

They have hijacked a *star chart* from a transmission from a human television, and have marked it appropriately. They represent the star chart as an ASCII grid with r rows and c columns, as follows:

- `.` represents empty space
- `#` represents a black hole that will kill them if they enter it
- `X` represents their current location, which would otherwise be empty space if they were not there
- A digit from `0` to `9` represents a star that they want to visit.
  - The *priority* of a star is equal to the numerical value of the digit that represents it on the star chart.
  - No two stars on the star chart have the same priority.

A command sequence is a string that contains only these letters, corresponding to the following movements for the ship. Executing the command sequence means performing each indicated movement, in order.

- `N` to move to the location directly north of your current location.
- `S` to move to the location directly south of your current location.
- `E` to move to the location directly east of your current location.
- `W` to move to the location directly west of your current location.

Here, "directly north/south/east/west" is informally understood to mean "one step" in the indicated direction.
The Grebulons wish to find a command sequence that satisfies these criteria:

- Obviously, they should not die. So do not enter any black holes! And do not exit the star chart (lest they fall off the edge of the universe).
- By executing this sequence, they should visit each cell that contains a star **exactly once** each (if they visit the same star twice, they die from overexposure to radiation).
- But they must visit the stars in the correct order: if Star α has a greater priority than Star β, then Star α should be visited *first*
- Also, they get bored easily, so none of the following should appear as a substring of the command sequence: `NN`, `SS`, `EE`, or `WW`

There may be many command sequences that satisfy these restrictions, so they also give these two additional criteria:

- The length of the command sequence should be *minimal*.

- Among all command sequences of minimal length, we choose the one that is lexicographically minimal (i.e. the one that would come first in the dictionary).

If the task is impossible, then that is also something that they need to know.
Oh man, that's so many restrictions! The Grebulons can't handle this. Maybe you could help them out instead?

Input
The first line contains a single integer T denoting the number of test cases.

Each test case begins with a single line containing two space-separated integers r and c, the number of rows and columns in the star chart. This is followed by r lines that each contain a string of length c, the star chart encoded in an ASCII grid as described in the problem statement.

- $1 \le T \le 1000$
- $1 \le r, c$
- $2 \le \sum rc \le 2 \times 10^5$ across all test cases
- Exactly one character in each star chart is `X`
- Each star chart contains at least one star, and all stars' priorities are distinct.

Output
For each test case, output the unique command sequence that satisfies the given criteria, or `IMPOSSIBLE` if no such command sequence exists.

input
4 4 3 0.. ... ... #X# 3 3 X.# 6.. 9.. 1 9 X.321#### 2 2 X# #1

output
NENWNW ESEWSWN IMPOSSIBLE IMPOSSIBLE

K. The Melancholy of Pollard's ρ

3 seconds, 256 megabytes

The Bistromath's Bistromathics Drive is malfunctioning! After taking a look at it, you and the crew have determined that this was the result of active sabotage.

The Bistromathics Drive is fueled by the very fundamental properties of numbers. One such fundamental property is the prime factorization of the integers (commonly expressed through the Fundamental Theorem of Arithmetic). The Bistromathics Drive relies on a subroutine that uses the Pollard's ρ algorithm in order to factor relatively large numbers

However, we noticed a flaw in our implementation. Pollard's Rho requires a pseudorandom sequence of nonnegative integers less than some n, which it typically generates with the following method:

- Let the *seed* s be an integer such that $0 \le s < n$.
- Define the function g as follows:

$$g(x) = (x^2 + a) \bmod n$$

  where in this implementation, a is an integer that was hard-coded into the algorithm.

- Then, repeatedly applying g to s generates a pseudorandom sequence. In other words, we consider the sequence $s, g(s), g(g(s)), g(g(g(s))), \ldots$

Here's the issue—it may be possible for $s, g(s), g(g(s)), g(g(g(s))), \ldots$ to not truly be random; in fact, more strongly, we can force this to be a **constant sequence**! If n and a are given, then we can maliciously select a seed s in order to force the pseudorandom sequence to be constant.

Help the team with debugging so that they can get the Bistromathics Drive working again. Given n and a, find **all** seeds s such that $0 \le s < n$ and $s = g(s) = g(g(s)) = g(g(g(s))) = \ldots$.

To make this problem a bit easier, **you are guaranteed that n is squarefree.**

Input
Input consists of a single line with two space-separated integers n and a.

Constraints

- $2 \le n \le 10^{18}$ and n is squarefree
- $0 < a < n$

Output
Let k be the number of seeds s that result in the process generating a constant sequence.

First, output a single line containing the integer k.

If $k \le 10^6$, then you should output k lines, containing each of the valid seeds in its own line, sorted in increasing order.

If $k > 10^6$, then instead, you output the line `ABSOLUTELY WAY TOO MANY` and nothing else

input
103 10

output
2 48 56

input
14 2

output
2 4 11

input
5 1

output
0