ATENEO DE MANILA UNIVERSITY

icpc international collegiate programming contest

icpc.foundation

Problem Set

Think.

Create.

Solve.

2019 Asia-Manila Regional
Programming Contest
14–15 December 2019
Ateneo de Manila University

JET BRAINS

DOST-PCIEERD

Smart acer Google

Contents

Notes

- Many problems have large input file sizes, so use fast I/O. For example:
  - In Java, use BufferedReader and PrintWriter.
  - In C/C++, use scanf and printf.
- All problems are solvable in C++, and Java, but the same is not guaranteed for Python due to its slowness.
- Good luck and enjoy the problems!

Very important! Read the following:

- Your solution will be checked by running it against several hidden test cases. You will not have access to these cases, but a correct solution is expected to handle them correctly.
- The output checker is **very strict**. Follow these guidelines strictly:
  - It is **space sensitive**. Do not output extra leading or trailing spaces. Do not output extra blank lines unless explicitly stated.
  - It is **case sensitive**. So, for example, if the problem asks for the output in lowercase, follow it.
  - Do not print any tabs. (No tabs will be required in the output.)
  - Do not output anything else aside from what's asked for in the Output section. So, do not print things like "Please enter t".

  Not following the output format strictly and exactly will likely result in a **Wrong answer** verdict.
- Do not read from, or write to, a file. You must read from the standard input and write to the standard output.
- For Java, do not add a package line at the top of your code. Otherwise, your submission will be judged **Runtime Error**.
- Only include one file when submitting: the source code (.cpp, .java, .py, etc.) and nothing else.
- Only use letters, digits and underscores in your filename. Do not use spaces, or other special symbols.
- Many problems have large input file sizes, so use fast I/O. For example:
  - In Java, use BufferedReader and PrintWriter.
  - In C/C++, use scanf and printf.

  We recommend learning and using these functions during the Practice Session.
- All problems are solvable in C++, and Java, but the same is not guaranteed for Python due to its slowness.
- Good luck and enjoy the contest!

Problem A

Suffix Three

Time Limit: 1 second

We just discovered a new data structure in our research group: a **suffix three**!

It's very useful for natural language processing. Given three languages and three suffixes, a suffix three can determine which language a sentence is written in.

It's super simple, 100% accurate, and doesn't involve advanced machine learning algorithms.

Let us tell you how it works.

- If a sentence ends with "`po`", the language is Filipino.
- If a sentence ends with "`desu`", or "`masu`", the language is Japanese.
- If a sentence ends with "`mnida`", the language is Korean.

Given this, we need you to implement a suffix three that can differentiate Filipino, Japanese, and Korean.

Oh, did I say three suffixes? I meant four.

Input Format

The first and only line of input contains the sentence.

Constraints

- The sentence has at least 1 and at most 1000 characters.
- The sentence consists only of lowercase English letters and spaces. There are no tabs.
- The sentence has no leading or trailing whitespace and no two consecutive spaces.
- The sentence ends with one of the suffixes mentioned above.

Note: Constraints are *guaranteed* to be true. You don't have to check them; you may simply assume them to be true.

Output Format

Print a single line containing either `FILIPINO`, `JAPANESE`, or `KOREAN` (all in uppercase), depending on the detected language.

Sample Input 1	Sample Output 1
kamusta po	FILIPINO

Sample Input 2	Sample Output 2
genki desu	JAPANESE

Sample Input 3	Sample Output 3
ohayou gozaimasu	JAPANESE

Sample Input 4	Sample Output 4
annyeong hashimnida	KOREAN

Sample Input 5	Sample Output 5
hajime no ippo	FILIPINO

Sample Input 6	Sample Output 6
bensamu no sentou houhou ga okama kenpo	FILIPINO

Sample Input 7	Sample Output 7
ang halaman doon ay sarisari singkamasu	JAPANESE

Sample Input 8	Sample Output 8
si roy mustang ay namamasu	JAPANESE

Problem B

Miss Punyverse

Time Limit: 3 seconds

The Oak has n nesting places, numbered 1 to n. Nesting place i is home to b_i bees and w_i wasps.

Some nesting places are connected by branches. We call two nesting places *adjacent* if there exists a branch between them. A *simple path* from nesting place x to y is given by a sequence $s_0, \ldots, s_p$ of distinct nesting places, where p is a non-negative integer, $s_0 = x$, $s_p = y$, and s_{i-1} and s_i are adjacent for each $i = 1, \ldots, p$. The branches of The Oak are set up in such a way that for any two pairs of nesting places x and y, there exists a unique simple path from x to y. Because of this, biologists and computer scientists agree that The Oak is in fact, a tree.

A *village* is a *nonempty* set V of nesting places such that for any two x and y in V, there exists a simple path from x to y whose intermediate nesting places all lie in V.

A set of villages $\mathscr{P}$ is called a *partition* if each of the n nesting places is contained in exactly one of the villages in $\mathscr{P}$. In other words, no two villages in $\mathscr{P}$ share any common nesting place, and altogether, they contain all n nesting places.

The Oak holds its annual Miss Punyverse beauty pageant. The two contestants this year are: Ugly Wasp and Pretty Bee. The winner of the beauty pageant is determined by voting, which we will now explain. Suppose $\mathscr{P}$ is a partition of the nesting places into m villages $V_1, \ldots, V_m$. There is a local election in each village. Each of the insects in this village vote for their favorite contestant. If there are more votes for Ugly Wasp than Pretty Bee, then Ugly Wasp is said to *win* in that village. Otherwise, Pretty Bee *wins*. Whoever wins in the most number of villages wins.

As it always goes with these pageants, bees always vote for the bee (which is Pretty Bee this year) and wasps always vote for the wasp (which is Ugly Wasp this year). Unlike their general elections, no one abstains in voting for Miss Punyverse as everyone takes it very seriously.

Mayor Waspacito, and his assistant Alexwasp, wants Ugly Wasp to win. He has the power to choose how to partition The Oak into exactly m villages. If he chooses the partition optimally, determine the maximum number of villages in which Ugly Wasp wins.

Input Format

The first line of input contains t, the number of test cases.

The first line of each test case contains two space-separated integers n and m. The second line contains n space-separated integers $b_1, b_2, \ldots, b_n$. The third line contains n space-separated integers $w_1, w_2, \ldots, w_n$. The next $n - 1$ lines describe the pairs of adjacent nesting places. In particular, the ith of them contains two space-separated integers x_i and y_i denoting the numbers of two adjacent nesting places.

Constraints

- $1 \leq t \leq 100$
- $1 \leq m \leq n \leq 3000$
- $0 \leq b_i, w_i \leq 10^9$
- $1 \leq x_i, y_i \leq n$

- It is guaranteed that The Oak forms a tree.
- The sum of the ns in a single file is $\leq 10^5$

Output Format

For each test case, output a single line containing a single integer denoting the maximum number of villages in which Ugly Wasp wins, among all partitions of The Oak into m villages.

Sample Input	Sample Output
2 4 3 10 160 70 50 70 111 111 0 1 2 2 3 3 4 2 1 143 420 214 349 2 1	2 0

Problem C

JaBloo 11: Lord of Expansion

Time Limit: 6 seconds

It is the year 20^{19}, and the most popular game among teenagers is JaBloo 11, a game made by Blinding Bear Blames. In this game, you play as a character chosen from one of c *character classes* (such as Paladin, Necromancer, or Pathfinder, among others), along with lots of other players playing in the shared game world. Your character starts at *level* 1 and you gradually increase your *level* by playing and defeating hordes of enemies. The level cap (or maximum possible level) is ℓ. It is guaranteed that:

- $100 \leq \ell \leq 10^5$. They originally believed that a low level cap of 99 grants the player a feeling of completion, but because the playerbase disagrees, they have now increased the level cap.
- $100 \leq c \leq 10^5$. They have added several character classes over several expansions of the game.

In their 357686312646216567629137th and latest expansion, *JaBloo 11: Lord of Expansion*, JaBloo has now become the Lord of Expansion, and he has expanded his control over the mortal world and caused many players to distrust and fight each other.

However, not every character can defeat any other character in battle. First, a character can only defeat other characters with a strictly smaller *level*. Furthermore, some *character classes* are weak to others. For example, a Druid would likely beat a Paladin in a PvP (player versus player) match. Exactly which character classes can defeat which character classes is determined by the mechanics of the game which is ultimately determined by the game developers.

For a pair of character classes (w, d), we say that character class w is *ascendant* to character class d if characters of class w can defeat characters of class d. Otherwise, we say that the pair (w, d) is *normal*. Note that:

- It is possible that $w = d$, that is, some characters can defeat other characters of the same class.
- It is possible that both w is ascendant to d and that d is ascendant to w. This usually happens when w and d are offense-oriented classes.
- It is possible that neither of two classes are ascendant to each other. This usually happens when the classes are defense-oriented.
- If w is ascendant to d and d is ascendant to d', it does not necessarily follow that w is ascendant to d'.

After extensive analysis, it has been determined which pairs of character classes are ascendant. There are k pairs of character classes that are ascendant, and the $c^2 - k$ remaining pairs are normal.

In summary, a character x can defeat a character y if and only if both the following are true:

- x has a strictly larger level than y, and
- the character class of x is ascendant to the character class of y.

JaBloo's personal assistant, JaRed, is helping him achieve his plans to kill as many characters as possible. There are n characters in the game world, and JaRed has the power to corrupt the minds and souls of some characters and instruct them to fight and defeat any other character of his choosing. However, his power is limited, and he can only corrupt the mind and soul of any particular character at most a times. In other words, JaRed can only cause a character to fight at most a other characters. Furthermore, JaRed cannot corrupt the minds of already-defeated characters.

If he plans his corruption carefully, what is the fewest number of people that can remain alive at the end? You may assume that the levels of the characters don't change throughout this process, and no battles occur except the ones caused by Jared's corruption.

Also, you are not sure what the value of a is (maybe JaRed has also corrupted your mind?), so you need to answer this question for each a from 1 to n.

Input Format

There is only one test case.

The first line of input contains four space-separated integers n, ℓ, c and k. The next n lines describe the characters. In particular, the ith following line contains two space-separated integers ℓ_i and c_i:

- ℓ_i denotes the level of character i.
- c_i is the character class of character i. For simplicity, we represent the character classes by positive integers between 1 and c.

The next k lines describe the ascendancies. In particular, the jth of the k following lines contains two space separated integers w_j and d_j denoting that character class w_j is ascendant to character class d_j.

Constraints

- $1 \leq n \leq 4200$
- $100 \leq \ell, c \leq 10^5$
- $1 \leq k \leq 32$
- $1 \leq \ell_i \leq \ell$
- $1 \leq c_i, w_j, d_j \leq c$
- Each ordered pair (w_j, d_j) appears at most once.
- w_j may equal d_j.

Output Format

Output n lines. The ith line must contain a single integer denoting the answer to the question for $a = i$.

Sample Input	Sample Output
4 124 100 3 114 1 11 1 114 100 124 1 1 1 1 100 80 100	2 1 1 1

(this page intentionally left almost blank)

Problem D

Okkeika Ferry Co.

Time Limit: 3 seconds

An archipelago is a group of islands. This archipelago actually has n islands, which we index $1, \ldots, n$. Between m pairs of these islands, there exist a ferry service. A pair (x, y) of islands with a ferry service means that there is a scheduled ferry which brings goods from island x directly to island y and a scheduled ferry which brings goods from island y to directly to island x. The rest of the pairs of islands, while not having a ferry service between them, may still be able to transfer or exchange goods with each other but the goods might have to stop in other islands.

Okkeika Ferry Co. is a company which determines which islands have ferry services between them. Other companies, called their clients, also send them requests to move goods around. At this moment, Okkeika Ferry Co. has k requests from their clients. A request from a client is a statement of the form: 'I have goods from island x that I want to transfer to island y'.

Some of these requests might be already possible using the currently existing services. However, some of these requests may be impossible to fulfill by exclusively using ferry services. When we say a request is possible to fulfill, we mean that it is possible to fulfill using only ferries.

The clients oblige Okkeika Ferry Co. to introduce a ferry service between EXACTLY one pair (a, b) of distinct islands which does not have a ferry service yet. Your job is to give a pair (a, b) of islands such that after adding a ferry service to the pair, the number of requests that can be fulfilled out of the k existing ones is maximized. If there are multiple options, output the lexicographically smallest one, as described in the output format section. (The pair (a_1, b_1) is lexicographically smaller than the pair (a_2, b_2) if either $a_1 < a_2$ or both $a_1 = a_2$ and $b_1 < b_2$.)

Even when the number of fulfilled requests cannot be increased by adding a ferry service to a pair, you must still add one anyway (for PR purposes)! The only time you are allowed not to add a ferry service is when it is impossible to do so. In which case, you should say so (so that the clients will be appeased).

Are you up for this challenge? Hmmmmmmmmmm?

Input Format

The first line of input contains t, the number of test cases.

The first line of each test case consists of a single integer n.

The second line contains a single integer m. The next m lines describe the ferry services. Each such line contains two space-separated integers x and y describing a ferry service existing between islands x and y.

The next line contains a single integer k. The next k lines describe the requests. Each such line contains two space-separated integers x and y describing a single request from a client of the form 'I have goods from island x that I want to transfer to island y'.

Constraints

- $1 \leq t \leq 50000$
- $1 \leq n, m, k \leq 10^5$

- $1 \leq x, y \leq n$
- $x \neq y$
- The sum of the ms in a single file is $\leq 2 \cdot 10^5$.
- The sum of the ks in a single file is $\leq 2 \cdot 10^5$.
- Every pair of islands has at most one ferry service between them.

Output Format

For each test case, output a single line according to the following:

- If it is impossible to add a ferry service, then print the string `NOT OKAY`.
- Otherwise, print three space-separated integers F, a and b. Here, F is the maximum number of requests that can be fulfilled, and (a, b) is the lexicographically smallest pair of distinct islands which does not have a ferry service yet such that adding a ferry service between them achieves this maximum.

Sample Input	Sample Output
1 6 3 1 4 2 5 3 6 4 6 1 2 1 1 5 4 5	3 1 2

Problem E

Do You Wanna Build More Snowmen?

Time Limit: 2 seconds

Olaf the Snowman has gotten a layer of permafrost around him since we last saw him. Now he's effectively immortal and worries about the notion that nothing is permanent. Because she is older and thus all-knowing, Elsa has decided to make more snowmen so that he can have friends so they can finish each other's sandwiches. For some unknown reason, Olaf wants to name the first one Samantha.

Elsa's power flurries from the air into the ground, so she makes snowballs in the sky that drop down to the top of *one tall tower*. Elsa has the ability to create multiple types of snowball. However, she doesn't have much control over her powers so she can't control the exact type of snowball she creates next. Fortunately she can just let it go and throw away the next snowball if she doesn't like it.

More specifically, there are 26 types of snowballs, and so we can represent them by uppercase English letters. The sequence of snowball types Elsa creates can be represented by a string s, where the ith letter, s_i, means that the ith snowball that Elsa creates is of type s_i. Elsa has no control over s_i, but at least she can decide to either let the ith snowball drop onto the tower or let it go and throw it away permanently. Elsa can only create at most $|s|$ snowballs. ($|s|$ denotes the length of string s.)

Elsa can also animate a contiguous section of the tower into a snowman. However, not all sections of snowballs form stable snowmen; there are only n designs that will form stable snowmen. The ith design can be represented by a string d_i representing the required sequence of snowball types, from bottom to top. If the sequence d_i is found somewhere in the tower contiguously, that part of the tower may be removed and animated into a snowman, and the snowballs above that section fall down into place. You may assume that the snowballs are sturdy enough and won't break.

Different designs of snowmen give Olaf different amounts of happiness. Specifically, the ith design gives Olaf p_i points of happiness. Elsa may create as many snowmen of any type as she wants (as long as it is possible).

Elsa doesn't want to go into the unknown unprepared and so she wondered: what is the maximum total happiness that Olaf can get if she builds snowmen optimally?

Input Format

The first line of input contains t, the number of test cases.

The first line contains a single string s representing the sequence of snowball types that Elsa creates. The ith character of this string, s_i, represents the type of the ith snowball Elsa creates.

The next line contains a single integer n. The n following lines describe the stable designs of snowmen. In particular, the ith line contains a string d_i (the design of the ith stable snowman), followed by a space, and then finally followed by an integer p_i denoting how many points of happiness this design gives Olaf.

Constraints

- $1 \leq t \leq 3$

- $1 \leq |s| \leq 120$
- $1 \leq n \leq 120$
- $1 \leq |d_i| \leq 120$
- $1 \leq |d_1| + |d_2| + \ldots + |d_n| \leq 120$
- $1 \leq p_i \leq 10^9$
- The d_is are distinct.
- All letters of s and d_i are uppercase English letters.

Output Format

For each test case, output a single line containing a single integer denoting the maximum possible total happiness points that Olaf can have.

Sample Input	Sample Output
2 WEEWOOO 3 EE 108 EO 107 OO 105 LIVESTREAMERS 8 TEAM 11 TREES 14 LIVES 16 LIVE 14 LIVER 20 VEST 19 VESTS 22 REAR 8	214 31

Explanation

In the first sample case, a maximum total happiness of 214 can be achieved. One way to do so is as follows:

- Initially, the tower of snowballs is empty.
- Create the 1st snowball, which is of type `W`, and let it fall onto the tower.

  From bottom to top, the tower is now: `W`.
- Create the 2nd snowball, which is of type `E`, and let it fall onto the tower.

  From bottom to top, the tower is now: `WE`.
- Create the 3rd snowball, which is of type `E`, and let it fall onto the tower.

  From bottom to top, the tower is now: `WEE`.
- Create the 4th snowball, which is of type `W`, but throw it away.

From bottom to top, the tower is now: `WEE`.

- Create the 5th snowball, which is of type `O`, and let it fall onto the tower.

  From bottom to top, the tower is now: `WEEO`.
- Create the 6th snowball, which is of type `O`, and let it fall onto the tower.

  From bottom to top, the tower is now: `WEEOO`.
- Animate a snowman with design `EO`, giving Olaf 107 points of happiness.

  From bottom to top, the tower is now: `WEO`.
- Animate a snowman with design `EO`, giving Olaf 107 points of happiness.

  From bottom to top, the tower is now: `W`.
- Create the 7th snowball, which is of type `O`, and let it fall onto the tower.

  From bottom to top, the tower is now: `WO`.

(this page intentionally left almost blank)

Problem F

Marking the Territory

Time Limit: 3 seconds

Spoopy-Doo is a dog. He marks his territory by urinating on trees. He plays around the rectangular forest, which we represent by an $r \times c$ grid. Each cell of the grid has exactly one tree. He has already marked his territory on n trees.

For our purposes, a *path* is a sequence of adjacent cells. One cell is *adjacent* to the other if it is either directly above, directly below, directly to the left or directly to the right of the other. The *length* of a path is the number of cells in the sequence. If the length of the path is odd, we define the *middle cell* of the path as the middle term of the sequence.

Spoopy-Doo plays a game by himself. In each turn of this game he chooses a pair of trees a and b which he has already marked. He then chooses a path P of *minimal length* between these two trees. If the path P has an even length, he does nothing. If the path P has an odd length, then he marks his territory on the middle cell of path P.

Determine if Spoopy-Doo can mark a target cell T within 4444 turns of the game he made up. If it is possible to mark T, find one sequence of moves that enables him to mark T.

Input Format

The first line of input contains t, the number of test cases.

The first line of each test case contains three space-separated integers r, c and n. The second line contains i_T and j_T denoting the row and column number of the target cell T. The next n lines describe the locations of the trees that Spoopy-Doo has already marked; the kth of them contains two integers i_k and j_k denoting the row and column numbers of the kth cell containing a marked tree.

Constraints

- $1 \leq t \leq 111$
- $1 \leq r, c \leq 10^9$
- $1 \leq n \leq 6$
- $1 \leq i_T, i_k \leq r$
- $1 \leq j_T, j_k \leq c$
- The (i_k, j_k) pairs are distinct.

Output Format

For each test case, print one line containing a string:

- `SPOOPY` if it is impossible for Spoopy-Doo to mark T.
- `DOO` if it is possible for Spoopy-Doo to mark T.

Furthermore, if it is possible, output a few more lines describing the sequence of moves that enables him to mark T. The first line of these must contain a nonnegative integer $m \leq 4444$, the number of moves. The kth of the next m lines describes the kth move. It must contain six space-separated integers $i_a, j_a, i_b, j_b, i_c, j_c$, where (i_a, j_a) and (i_b, j_b) are the coordinates (row and

column number) of cells containing previously-marked trees, and (i_c, j_c) are the coordinates of the next cell to be marked. These cells must satisfy Spoopy-Doo's requirement described above; in particular, cell (i_c, j_c) must be the middle cell of some minimal-length path between cells (i_a, j_a) and (i_b, j_b). (Such a path necessarily has an odd length.)

There may be multiple possible answers; any one will be accepted.

Sample Input	**Sample Output**
3 11 11 2 6 6 5 1 5 11 1000 1000 3 7 6 5 5 5 6 11 5 1000 1000 1 123 123 123 123	SPOOPY DOO 2 11 5 5 5 8 5 5 6 8 5 7 6 DOO 0

Problem G

Guizzmo and the Computer Scientist's Stone

Time Limit: 3 seconds

After a long night of picking locks, incapacitating guards, and doing dynamic warm-up stretches, master thief Guizzmo is finally in the same room as the legendary *Computer Scientist's Stone*, or CSS, said to be able to provide polynomial-time reductions between any two algorithmic problems.

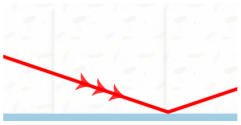
However, Guizzmo has one final hurdle before he can make P=NP: the *laser grid* protecting the CSS. Luckily, Guizzmo has a gizmo that can remotely deactivate individual laser beams. However, it takes time to charge up his gizmo, and he wants to minimise the time before he can steal the stone. Help him find out the minimum number of laser beams he has to deactivate before he can reach the stone!

You will be given the specifications of the room, which is in the shape of a *simple rectilinear polygon* (all of its edges are horizontal or vertical; and the edges do not intersect each other except at the vertices of succeeding edges). Some of the walls of the room are *mirrors*.

You will also be given Guizzmo's current location, and the (different) location of the CSS. It is guaranteed that Guizzmo and the CSS are strictly in the interior of the room, and are not directly on a wall or corner of the room.

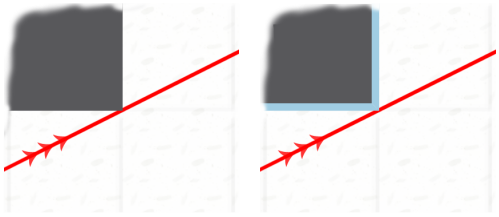
In the room, there are ℓ *laser sources* installed. Each laser source is installed on a wall of the room, and it is guaranteed that no laser source is installed on a corner. Each laser source is oriented in a fixed direction, which is neither perfectly horizontal nor perfectly vertical. Each laser source is pointed towards the interior of the room. If a laser hits a non-mirror wall, it dissipates (disappears). If a laser hits a mirror, it is reflected at an angle equal to its angle of incidence.

(a) A laser hitting a non-mirror wall dissipates.

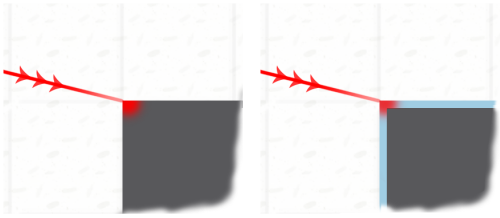


(b) A laser hitting a mirror will be reflected.

If a laser grazes a corner, such that it can continue in a straight line within the interior of the room, it continues in its path; however, if a laser hits a corner directly, it dissipates. These happen even if one or both of the walls forming the corner are mirrors.



(a) A laser *grazing* a corner continues in its path.



(b) A laser *directly hitting* a corner dissipates.

Lasers do not interfere with each other, neither constructively nor destructively. If a laser directly passes through Guizzmo's current location or the CSS's location, it must be deactivated so

he can reach the Stone.

What is the minimum number of laser sources that must be deactivated in order for there to be a continuous path from Guizzmo to the CSS that does not intersect any laser?

Input Format

There is only one test case.

The first line of input contains two space-separated integers n and ℓ denoting the number of vertices of the polygon (representing the room's shape) and the number of laser sources, respectively. The next n lines describe the coordinates of the vertices of the polygon. In particular, the ith following line contains two space-separated integers x_i and y_i denoting that (x_i, y_i) is a vertex of the polygon. The vertices are given in either clockwise or counterclockwise order around the polygon.

The next line contains a string of n characters, the ith of which is either `M` or `W` denoting that the side of the polygon between vertices (x_i, y_i) and (x_{i+1}, y_{i+1}) is either a `M`irror or a `W`all, respectively. (We identify $(x_{n+1}, y_{n+1}) = (x_1, y_1)$.)

The next ℓ lines describe the laser sources. In particular, the ith of these lines contains four space-separated integers $x_{i,\text{pt}}, y_{i,\text{pt}}, x_{i,\text{dir}}, y_{i,\text{dir}}$, denoting that the ith laser source is located at $(x_{i,\text{pt}}, y_{i,\text{pt}})$ and points towards the direction specified by the vector $\langle x_{i,\text{dir}}, y_{i,\text{dir}} \rangle$.

The next line contains two space-separated integers x_G and y_G denoting that (x_G, y_G) is Guizzmo's current location.

The next (and final) line contains two space-separated integers x_S and y_S denoting that (x_S, y_S) is the location of the CSS.

Constraints

- $1 \leq n \leq 100$
- $1 \leq \ell \leq 30$
- $|x_i|, |y_i| \leq 30$
- $1 \leq |x_{i,\text{dir}}|, |y_{i,\text{dir}}| \leq 10$
- The polygon is rectilinear, is simple, and does not contain straight angles.
- Each laser source is along the perimeter but never exactly on a vertex of the polygon.
- Each laser source is guaranteed to point inward the polygon.
- Each laser never points perfectly horizontally or vertically.
- The locations of Guizzmo and the CSS are inside the polygon and not on the boundary.
- The locations of Guizzmo and the CSS are distinct.

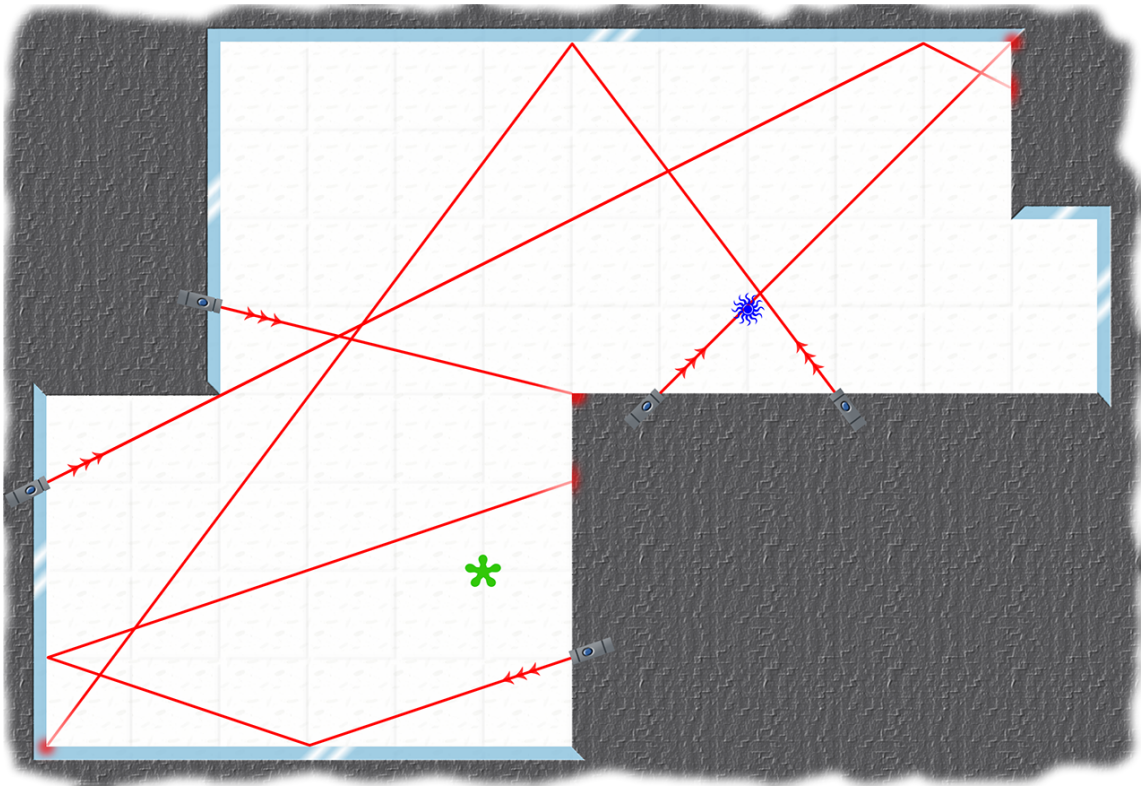
Output Format

Output a single line containing a single integer denoting the minimum number of laser sources that must be deactivated in order for there to be a continuous path from Guizzmo to the CSS that does not intersect any laser.

Sample Input	Sample Output
10 5 -4 0 -4 4 5 4 5 2 6 2 6 0 0 0 0 -4 -6 -4 -6 0 MMWMMWWMMW 3 0 -6 8 1 0 3 3 0 -3 -3 -1 -6 -1 2 1 -4 1 8 -2 -1 -2 2 1	3

Explanation

The following image illustrates the sample test case. The red lines represent lasers. The blue walls are the walls with mirrors. The green spot is Guizzmo's current location, and the blue spot is the location of the CSS.



Guizzmo can deactivate three lasers so that he can reach the CSS. It should be easily seen that deactivating two lasers is not enough.

(this page intentionally left almost blank)

Problem H

Kirchhoff's Current Loss

Time Limit: 2 seconds

Your friend Kirchhoff is shocked with the current state of electronics design.

"Ohmygosh! Watt is wrong with the field? All these circuits are inefficient! There's so much capacity for improvement. The electrical engineers must not conduct their classes very well. It's absolutely revolting," he said.

The negativity just keeps flowing out of him, but even after complaining so many times he still hasn't lepton the chance to directly change anything.

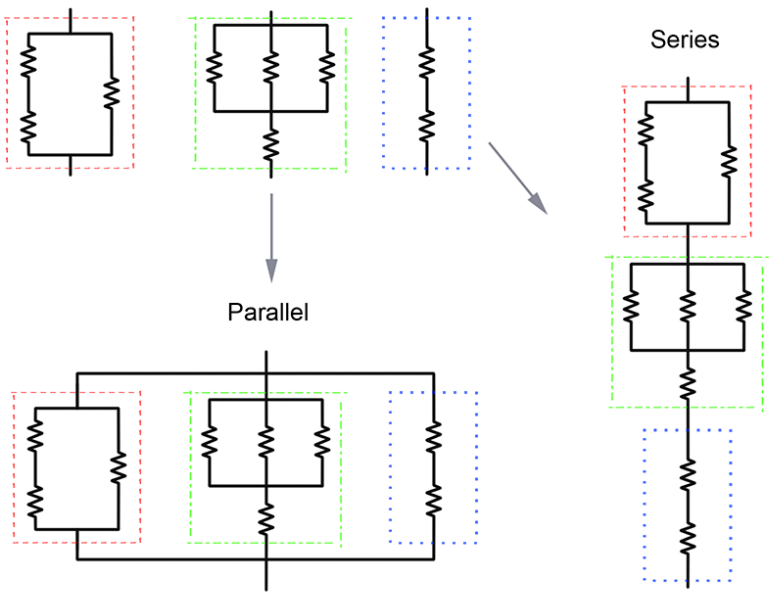
"These circuits have too much total resistance. Wire they designed this way? It's just causing a massive loss of resistors! Their entire field could conserve so much money if they just maximized the potential of their designs. Why can't they just try alternative ideas?"

The frequency of his protests about the electrical engineering department hertz your soul, so you have decided to take charge and help them yourself. You plan to create a program that will optimize the circuits to find the minimum total resistance required while keeping the same circuit layout and maintaining the same effective resistance.

A *circuit* has two endpoints. If we apply a voltage of V between these two endpoints, then some amount of current I is generated. These two values are related via the equation $V = IR$, where R is a constant depending only on the circuit, and it is called its **effective resistance**.

The circuits we'll consider will be formed from a number of simple resistors, joined together in *series* or in *parallel*, forming more complex circuits. These circuits may then again be joined together in series or parallel, forming even more complex circuits, and so on.

The following image illustrates combining circuits in series or parallel.



According to your friend Kirchhoff, the effective resistance can be calculated quite easily when joining circuits in series or in parallel:

- When joining k circuits in *series* with effective resistances $R_1, R_2, \ldots, R_k$, the effective resistance R of the resulting circuit is the sum

$$R = R_1 + R_2 + \ldots + R_k.$$

- When joining k circuits in *parallel* with effective resistances $R_1, R_2, \ldots, R_k$, the effective resistance R of the resulting circuit is found by solving for R in

$$1/R = 1/R_1 + 1/R_2 + \ldots + 1/R_k,$$

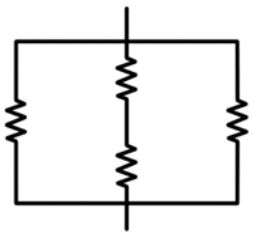
assuming all $R_i > 0$; if there is at least one component that has effective resistance 0, then the effective resistance of the whole circuit is simply $R = 0$.

Since the circuits we'll consider are formed by joining smaller circuits in series or parallel, their effective resistances can be computed recursively.

To make things easier to parse, circuits will be represented by strings. (Well, if you want us to give the input as an ASCII drawing, let us know!) Individual resistors are simply represented by an asterisk, `*`. More complex circuits are formed by joining the string representations of smaller circuits, as follows: Suppose $s_1, s_2, \ldots, s_k$ are the string representations of k circuits (where $k \geq 2$). Then:

- the representation of their *series* circuit is `(`s_1 `S` s_2 `S ... S` s_k`)`.
- the representation of their *parallel* circuit is `(`s_1 `P` s_2 `P ... P` s_k`)`.

For example, the string `(* P (* S *) P *)` represents the following circuit:



Given the string representation of a circuit, your task is to assign the resistances of the individual resistors such that they satisfy the following requirements:

- Each individual resistor has a *nonnegative integer* resistance value,
- The effective resistance of the whole circuit is r, and
- The sum of the resistances of the individual resistors is minimized.

You need to output the string representation of that circuit, but with each asterisk (`*`) replaced by the assigned resistance of that resistor. If it is impossible to accomplish the task, you must say so as well.

Input Format

The first line of input contains t, the number of test cases.

Each test case consists of a single line containing the integer r, a space, and then the string representation of the circuit. It is guaranteed that the string representation is valid and follows the description above.

Constraints

- $1 \leq t \leq 32000$
- The number of $\star$ in a single case is at least 1 and at most 80000.
- The number of $\star$ in a single file is at most 320000.
- $1 \leq r \leq 10^6$

Output Format

For each test case, print a single line:

- If it's possible to achieve an effective resistance of r, then print `REVOLTING!` (including the exclamation mark), then a space, then the string representation of the same circuit but with the asterisks replaced by their assigned resistances, such that the sum of the resistances of the individual resistors is minimized.

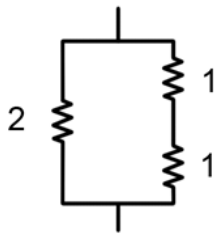
  There may be multiple possible such assignments; any one will be accepted.

- If it's impossible, print the exact string `KIRCHHOFF'S LOSS` (including the apostrophe)

Sample Input	Sample Output
3 5 * 1 (* S *) 1 (* P (* S *))	REVOLTING! 5 REVOLTING! (1 S 0) REVOLTING! (2 P (1 S 1))

Explanation

The following illustrates the circuit for the third sample case:



Here, the sum of the resistances of the individual resistors is $2+1+1=4$, which can be shown to be the minimum. Note that there may be other assignments that achieve this minimum.

(this page intentionally left almost blank)

Problem I

A Case By Case Basis

Time Limit: 3 seconds

You are given two versions of a letter of the English alphabet. One of them is its lowercase version (small letter) and the other is its uppercase version (big letter).

Your goal is to determine which of the two versions is the uppercase version. Oh, and to make this problem more accessible to people with vision problems, we write each of the letters using a 9×5 array of characters so that you can see it clearly.

The exact representations of all 26 uppercase and lowercase English letters will be given in a page after the sample input/output section. The letters there are written in alphabetical order, with each uppercase letter appearing right before its corresponding lowercase version. Each pixel will be represented by either `#` for **black**, or `.` for **white**. For example, here is how we represent the uppercase `H` and lowercase `h`, respectively:

```
#...# #....
#...# #....
#...# #....
##### #.##.
#...# ##..#
#...# #...#
#...# #...#
..... .....
..... .....
```

Input Format

The first line of input contains t, the number of test cases.

Each test case will consist of 9 lines, with 11 characters each. The first 5 characters of these lines represent the rows of the first letter, and the last 5 of each line are the rows of second letter. Each character will either be `.` or `#`, except for the middle column which will all be spaces.

There is a blank line after each test case.

Constraints

- $1 \leq t \leq 100$
- Each test case will be valid. That is, for each test case, the input is guaranteed to represent the two versions of the same letter.

Output Format

Output a string of length t, where each character is either `1` of `2`. Specifically, for each i from 1 to t, the ith character must be:

- `1` if the first letter of the ith test case is the one in uppercase.
- `2` if the second letter of the ith test case is the one in uppercase.

Sample Input	Sample Output
2 #...# #.... #...# #.... #...# #.... ##### #.##. #...# ##..# #...# #...# #...# #...# ....#.. .###. ....#.. ..#.. ..#.. ..#.. ..#.. ..#.. ..#.. ..#.. ..#.. ..#.. .###.	12

Here are the representations of all uppercase and lowercase English letters:

(this page intentionally left almost blank)

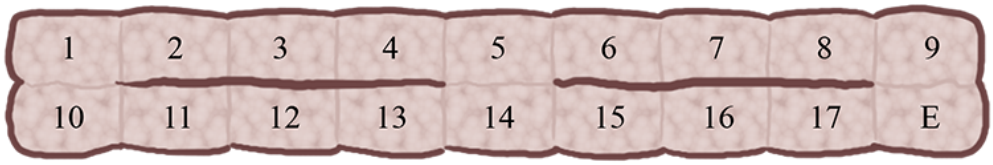
Problem J

Intergalactic Sliding Puzzle

Time Limit: 2 seconds

You are an intergalactic surgeon and you have an alien patient. For the purposes of this problem, we can and we will model this patient's body using a $2 \times (2k+1)$ rectangular grid. The alien has $4k+1$ distinct organs, numbered 1 to $4k+1$.

In healthy such aliens, the organs are arranged in a particular way. For example, here is how the organs of a healthy such alien would be positioned, when viewed from the top, for $k = 4$:



Here, the E represents empty space.

In general, the first row contains organs 1 to $2k+1$ (in that order from left to right), and the second row contains organs $2k+2$ to $4k+1$ (in that order from left to right) and then empty space right after.

Your patient's organs are complete, and inside their body, but they somehow got shuffled around! Your job, as an intergalactic surgeon, is to put everything back in its correct position. All organs of the alien must be in its body during the entire procedure. This means that at any point during the procedure, there is exactly one cell (in the grid) that is empty. In addition, you can only move organs around by doing one of the following things:

- You can switch the positions of the empty space E with any organ to its immediate left or to its immediate right (if they exist). In reality, you do this by sliding the organ in question to the empty space.
- You can switch the positions of the empty space E with any organ to its immediate top or its immediate bottom (if they exist) only if the empty space is on the *leftmost* column, *rightmost* column or in the *centermost* column. Again, you do this by sliding the organ in question to the empty space.

Your job is to figure out a sequence of moves you must do during the surgical procedure in order to place back all $4k+1$ internal organs of your patient in the correct cells. If it is impossible to do so, you must say so. In this way, the alien's spouse and kids will be informed that they are now widowed and orphaned, respectively.

Input Format

The first line of input contains t, the number of test cases.

Each test case consists of three lines. The first line contains a single integer k which determines the size of the grid. The next two lines each contains $2k+1$ space-separated integers or the letter E, describing the first and second rows of organs, respectively.

Constraints

- $1 \leq t \leq 20$

- $1 \leq k \leq 20$
- There is exactly one `E`.
- It is guaranteed that all $4k + 1$ organs are present.

Output Format

For each test case, first print a single line containing either:

- `SURGERY COMPLETE` if it is possible to place back all internal organs in the correct locations.
- `SURGERY FAILED` if it is impossible.

If it is impossible, then this is the only line of output for the test case. However, if it is possible, output a few more lines describing the sequence of moves to place the organs in the correct locations.

The sequence of moves will be a string of letters `u`, `d`, `l` or `r`, representing sliding the organ that's directly above, below, to the left or to the right of the empty space, respectively, into the empty space. Print the sequence of moves in the following line, as such a string.

For convenience, you may use **shortcuts** to reduce the size of your output. You may use uppercase letters as shortcuts for sequences of moves. For example, you could choose `T` to represent the string `lddrr`. These shortcuts may also include other shortcuts on their own! For example, you could choose `E` to represent `TruT`, etc.

You may use any number of uppercase letters (including none) as shortcuts. The only requirements are the following:

- The total length of all strings in your output for a single case is at most 10^4.
- The resulting sequence of moves is finite, after expanding all shortcuts. Thus, there must be no cycles involving the shortcuts. Note that the final sequence of moves (after expanding) may be much longer than 10^4; the only requirement is that it's finite.

As an example, if `T = lddrr`, `E = TruT` and `R = rrr`, then `TurTlER` expands to:

- `TurTlER`
- **`lddrr`**`urTlER`
- `lddrrur`**`lddrr`**`lER`
- `lddrrurlddrrl`**`TruT`**`R`
- `lddrrurlddrrl`**`lddrr`**`ruTR`
- `lddrrurlddrrllddrrru`**`lddrr`**`R`
- `lddrrurlddrrllddrrrulddrr`**`rrr`**

To use shortcuts, print each one of them in a single line as the uppercase letter, a space, and then the string that this shortcut represents. They may be printed in any order. At the end of all of those, print a single line containing `DONE`. **Note:** You still need to print `DONE` even if you don't plan on using shortcuts.

Your sequence does not need to be the shortest. Any valid sequence of moves (satisfying the requirements above) will be accepted.

Sample Input

```
2
3
1 2 3 5 6 E 7
8 9 10 4 11 12 13
11
34 45 6 22 16 43 38 44 5 4 41 14 7 29 28 19 9 18 42 8 17 33 1
E 15 40 36 31 24 10 2 21 11 32 23 30 27 35 25 13 12 39 37 26 20 3
```

Sample Output

```
SURGERY COMPLETE
IR
R SrS
S rr
I lldll
DONE
SURGERY FAILED
```

Explanation

There are three shortcuts defined in the first sample output:

- `R = SrS`,
- `S = rr`, and
- `I = lldll`

The sequence of moves is `IR` and it expands to:

- `IR`
- **`lldll`**`R`
- `lldll`**`SrS`**
- `lldll`**`rr`**`rS`
- `lldllrrr`**`rr`**

(this page intentionally left almost blank)

Problem K

Cut and Paste

Time Limit: 2 seconds

We start with a string S consisting only of the digits 1, 2, or 3. The length of S is denoted $|S|$. For each i from 1 to $|S|$, the ith character of S is denoted S_i.

There is one cursor. The cursor's location, ℓ, is denoted by an integer in $\{0, \ldots, |S|\}$, with the following meaning:

- If $\ell = 0$, then the cursor is located before the first character of S.
- If $\ell = |S|$, then the cursor is located right after the last character of S.
- If $0 < \ell < |S|$, then the cursor is located between S_ℓ and $S_{\ell+1}$.

We denote by S_{left} the string to the left of the cursor, and S_{right} the string to the right of the cursor.

We also have a string C, which starts out as empty. Three actions are available.

- **The Cut action**. Set $C \leftarrow S_{\text{right}}$, then set $S \leftarrow S_{\text{left}}$.
- **The Paste action**. Append the value of C to the string S.
- **The Move action**. Move the cursor one step to the right. This increments ℓ once.

The cursor initially starts at $\ell = 0$. Then, we perform the following actions:

1. Perform the Move action once.
2. Perform the Cut action once.
3. Perform the Paste action S_ℓ times.
4. If $\ell = |S|$, stop. Otherwise, return to step 1.

You are given two integers x and n. Consider the time when ℓ first reaches the value x; this happens after some Move operation (step 1). What will be the last n characters of S_{left} during this exact moment? (Remember that S_{left} is the string to the left of the cursor.)

It is guaranteed that ℓ reaches x at some point.

Input Format

The first line of input contains t, the number of test cases.

The first line of each test case contains two space-separated integers x and n. The second line of each test case consists of the initial string S.

Constraints

- $1 \leq t \leq 250$
- $1 \leq |S| \leq 200$
- $n \leq x \leq 10^{16}$
- $1 \leq n \leq 200$

Output Format

For each test case, output a single line containing a single string: the required n characters.

Sample Input	Sample Output
3 7 3 2323 20 10 333 24 11 132133	323 3333333333 31332133213

Explanation

Let's illustrate what happens with the third test case. Initially, we have $S = \texttt{132133}$. Initially, $\ell = 0$ and $C = \varepsilon$ (the empty string). The following things happen if we follow the steps above:

- *Step 1, Move once:* we get $\ell = 1$.
- *Step 2, Cut once:* we get $S = \texttt{1}$ and $C = \texttt{32133}$.
- *Step 3, Paste S_ℓ = 1 time:* we get $S = \texttt{132133}$.
- *Step 4*: $\ell = 1 \neq |S| = 6$, so we return to step 1.
- *Step 1, Move once:* we get $\ell = 2$.
- *Step 2, Cut once:* we get $S = \texttt{13}$ and $C = \texttt{2133}$.
- *Step 3, Paste S_ℓ = 3 times:* we get $S = \texttt{13213321332133}$.
- *Step 4*: $\ell = 2 \neq |S| = 14$, so we return to step 1.
- *Step 1, Move once:* we get $\ell = 3$.
- *Step 2, Cut once:* we get $S = \texttt{132}$ and $C = \texttt{13321332133}$.
- *Step 3, Paste S_ℓ = 2 times:* we get $S = \texttt{1321332133213313321332133}$.
- *Step 4*: $\ell = 3 \neq |S| = 25$, so we return to step 1.
- *Step 1, Move once:* we get $\ell = 4$.
- *Step 2, Cut once:* we get $S = \texttt{1321}$ and $C = \texttt{3321332133213313321332133}$.
- *Step 3, Paste S_ℓ = 1 time:* we get $S = \texttt{1321332133213313321332133}$.
- *Step 4*: $\ell = 4 \neq |S| = 25$, so we return to step 1.
- *Step 1, Move once:* we get $\ell = 5$.
- and so on...

At some point, ℓ will reach $x = 24$, and S will be some (long) string. You can verify that at that exact time, the last $n = 11$ characters of S to the left of the cursor will be $\texttt{31332133213}$.

Problem L

Jeremy Bearimy

Time Limit: 2 seconds

Welcome! Everything is fine.

You have arrived in The Medium Place, the place between The Good Place and The Bad Place. You are assigned a task which will either make people happier, or torture them for eternity.

You have a list of k pairs of people which have arrived in a new inhabited neighborhood. You need to assign each of the $2k$ people into one of the $2k$ houses. Each person will be the resident of one house, and each house will only have one resident.

Of course, in the neighborhood, it is possible to visit friends. There are $2k-1$ roads, each of which connects two houses. It takes some time to traverse a road. We will specify the amount of time it takes in the input. The neighborhood is designed in such a way that from anyone's house, there is exactly one sequence of distinct roads you can take to any other house.

The truth is, these k pairs of people are actually soulmates. We index them from 1 to k. Let s be a pair of soulmates. We denote by $f(s)$ the amount of time it takes for them to go to each other's houses.

As we have said before, you will need to assign each of the $2k$ people into one of the $2k$ houses. You have two missions, one from the entities in The Good Place and one from the entities of The Bad Place. Here they are:

- The first mission, from The Good Place, is to assign the people into the houses such that the sum of $f(s)$ over all pairs s is minimized. We call this minimized sum G. This makes sure that soulmates can easily and efficiently visit each other.
- The second mission, from The Bad Place, is to assign the people into the houses such that the sum of $f(s)$ over all pairs s is maximized. We call this maximized sum B. This makes sure that soulmates will have a difficult time to visit each other.

What are the values of G and B?

Input Format

The first line of input contains t, the number of test cases.

The first line of each test case contains a single integer k denoting the number of pairs of people. The next $2k-1$ lines describe the roads; the ith of them contains three space-separated integers a_i, b_i, t_i which means that the ith road connects the a_ith and b_ith houses with a road that takes t_i units of time to traverse.

Constraints

- $1 \leq t \leq 500$
- $1 \leq k \leq 10^5$
- $1 \leq a_i, b_i \leq 2k$
- $1 \leq t_i \leq 10^6$
- The sum of the ks in a single test file is $\leq 3 \cdot 10^5$

Output Format

For each test case, output a single line containing two space-separated integers G and B.

Sample Input	Sample Output
1 3 1 2 3 3 2 4 2 4 3 4 5 6 5 6 5	15 33

Problem M

Beingawesomeism

Time Limit: 1 second

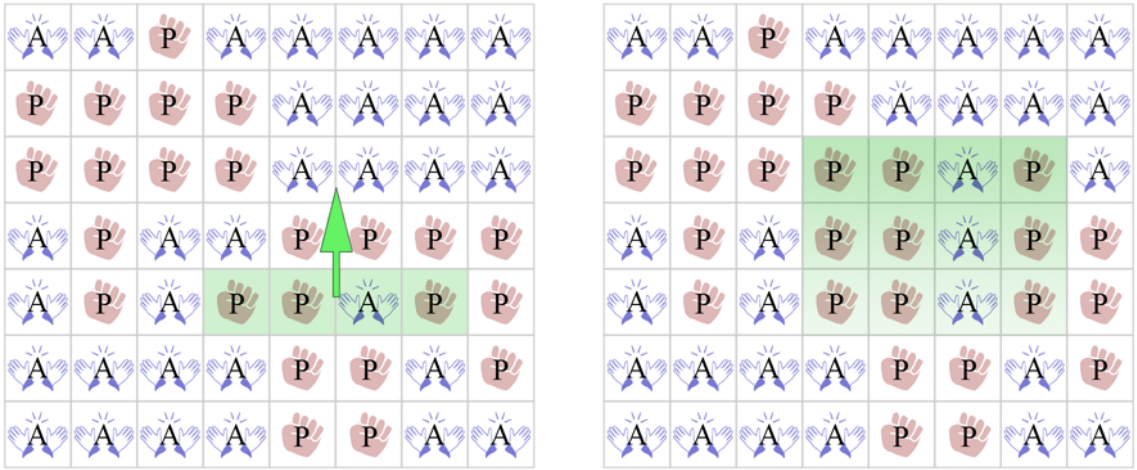
You are an all-powerful being and you have created a rectangular world. In fact, your world is so bland that it could be represented by a $r \times c$ grid. Each cell on the grid represents a country. Each country has a dominant religion. There are only two religions in your world. One of the religions is called Beingawesomeism, who do good for the sake of being good. The other religion is called Pushingittoofarism, who kill animals and other people and offer them to you.

Oh, and you are actually not really all-powerful. You just have one power, which you can use infinitely many times! Your power involves missionary groups. When a missionary group of a certain country, say a, passes by another country b, they change the dominant religion of country b to the dominant religion of country a.

In particular, a single use of your power is this:

- You choose a horizontal $1 \times x$ subgrid or a vertical $x \times 1$ subgrid. That value of x is up to you.
- You choose a direction d. If you chose a horizontal subgrid, your choices will either be NORTH or SOUTH. If you choose a vertical subgrid, your choices will either be EAST or WEST.
- You choose the number s of steps.
- You command each country in the subgrid to send a missionary group which will travel s steps towards direction d. In each step, they will visit (and in effect convert the dominant religion of) all s countries they pass through, as detailed above.

The following image illustrates one possible single usage of your power. Here, `A` represents a country with dominant religion Beingawesomeism and `P` represents a country with dominant religion Pushingittoofarism.



You are a being which believes in free will, for the most part. However, you just really want to stop receiving animal offerings and murders which are attributed to your name. Hence, you decide to use your powers and try to make Beingawesomeism the dominant religion in every country.

What is the minimum number of usages of your power needed to convert everyone to Beingawesomeism?

With god, nothing is impossible. But maybe you're not god? If it is impossible to make Beingawesomeism the dominant religion in all countries, you must also admit your mortality and say so.

Input Format

The first line of input contains t, the number of test cases.

The first line of each test case contains two space-separated integers r and c denoting the dimensions of the grid. The next r lines each contains c characters describing the dominant religions in the countries. In particular, the jth character in the ith line describes the dominant religion in the country at the cell with row i and column j, where:

- `A` means that the dominant religion is Beingawesomeism.
- `P` means that the dominant religion is Pushingittoofarism.

Constraints

- $1 \leq t \leq 2 \cdot 10^4$
- $1 \leq r, c \leq 60 \times 60$
- The sum of the rc in a single test file is $\leq 3 \cdot 10^6$
- The grid will only contain `A`s or `P`s.

Output Format

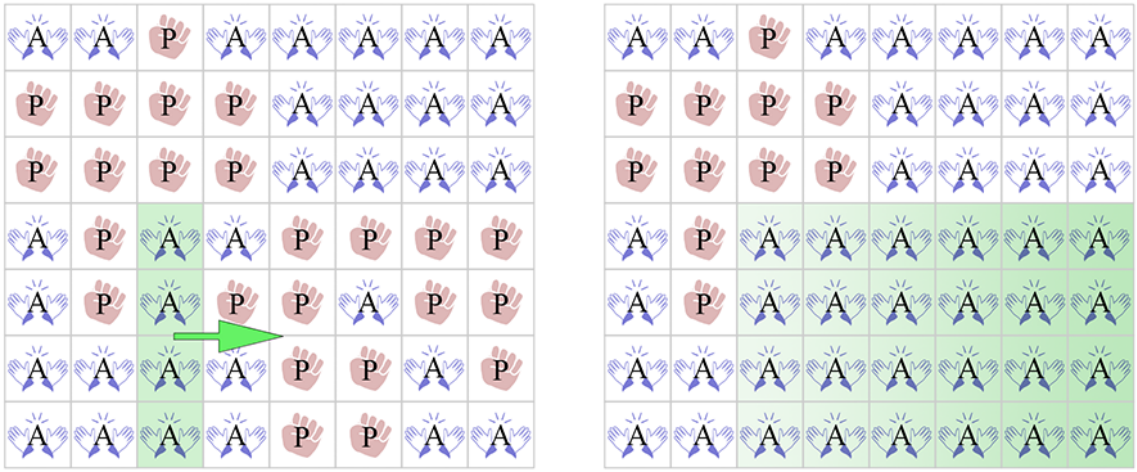
For each test case, output a single line containing the minimum number of usages of your power needed to convert the everyone to Beingawesomeism, or the string `MORTAL` if it is impossible to do so.

Sample Input	Sample Output
1 7 8 AAPAAAAA PPPPAAAA PPPPAAAA APAAPPPP APAPPAPP AAAAPPAP AAAAPPAA	2

Explanation

It can be done in two usages, as follows:

Usage 1:



Usage 2: