A. Cut and Paste

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

We start with a string s consisting only of the digits 1, 2, or 3. The length of s is denoted by $|s|$. For each i from 1 to $|s|$, the i-th character of s is denoted by 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 we call our clipboard, which starts out as empty. There are three types of actions:

- **The Move action**. Move the cursor one step to the right. This increments ℓ once.
- **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 end of the string s. Note that this doesn't modify c.

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

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

You're given the initial string s and the integer x. What is the length of s when the procedure stops? Since this value may be very large, only find it modulo $10^9 + 7$.

It is guaranteed that $\ell \leq |s|$ at any time.

Input

The first line of input contains a single integer t ($1 \leq t \leq 1000$) denoting the number of test cases. The next lines contain descriptions of the test cases.

The first line of each test case contains a single integer x ($1 \leq x \leq 10^6$). The second line of each test case consists of the initial string s ($1 \leq |s| \leq 500$). It is guaranteed, that s consists of the characters "1", "2", "3".

It is guaranteed that the sum of x in a single file is at most 10^6. It is guaranteed that in each test case before the procedure will stop it will be true that $\ell \leq |s|$ at any time.

Output

For each test case, output a single line containing a single integer denoting the answer for that test case modulo $10^9 + 7$.

Example

input
4 5 231 7 2323 6 333 24 133321333
output
25 1438 1101 686531475

Note

Let's illustrate what happens with the first test case. Initially, we have $s = 231$. Initially, $\ell = 0$ and $c = \varepsilon$ (the empty string). The following things happen if we follow the procedure above:

- Step 1, *Move once:* we get $\ell = 1$.
- Step 2, *Cut once:* we get $s = 2$ and $c = 31$.
- Step 3, *Paste s_ℓ = 2 times:* we get $s = 23131$.
- Step 4: $\ell = 1 \neq x = 5$, so we return to step 1.

- Step 1, *Move once:* we get $\ell = 2$.
- Step 2, *Cut once:* we get $s = 23$ and $c = 131$.
- Step 3, *Paste s_ℓ = 3 times:* we get $s = 23131131131$.
- Step 4: $\ell = 2 \neq x = 5$, so we return to step 1.

- Step 1, *Move once:* we get $\ell = 3$.
- Step 2, *Cut once:* we get $s = 231$ and $c = 31131131$.
- Step 3, *Paste s_ℓ = 1 time:* we get $s = 23131131131$.
- Step 4: $\ell = 3 \neq x = 5$, so we return to step 1.

- Step 1, *Move once:* we get $\ell = 4$.
- Step 2, *Cut once:* we get $s = 2313$ and $c = 1131131$.
- Step 3, *Paste s_ℓ = 3 times:* we get $s = 2313113113111311311131131$.
- Step 4: $\ell = 4 \neq x = 5$, so we return to step 1.

- Step 1, *Move once:* we get $\ell = 5$.
- Step 2, *Cut once:* we get $s = 23131$ and $c = 13113111311311131131$.
- Step 3, *Paste s_ℓ = 1 times:* we get $s = 2313113113111311311131131$.
- Step 4: $\ell = 5 = x$, so we stop.

At the end of the procedure, s has length 25.

B. Beingawesomeism

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

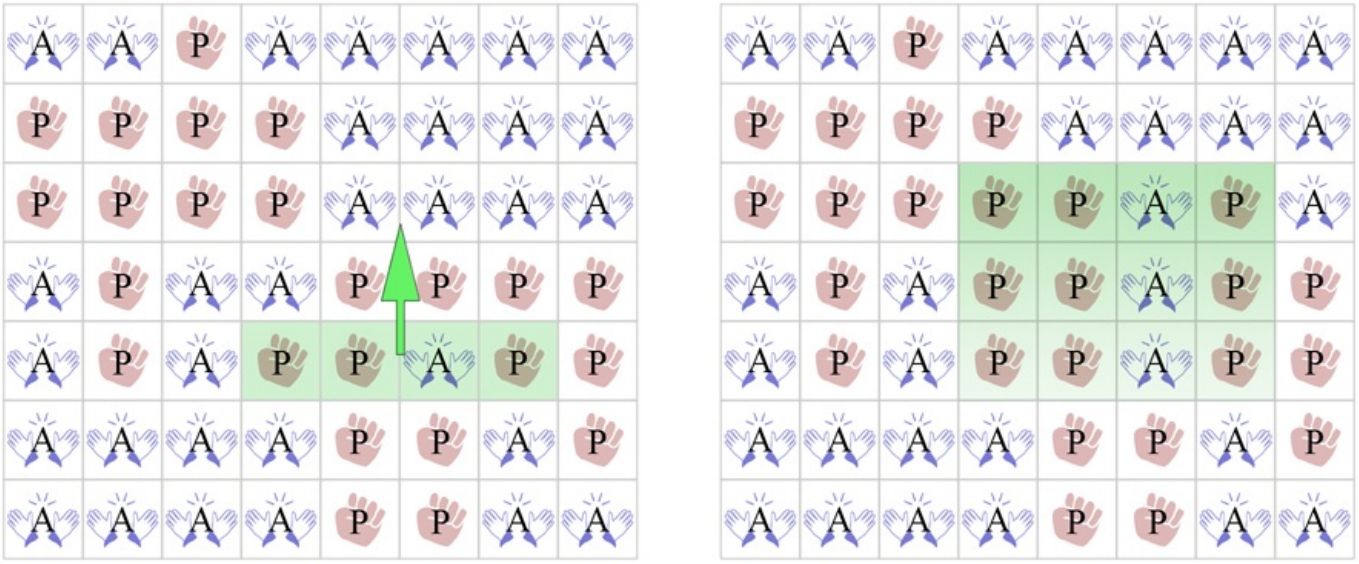
You are an all-powerful being and you have created a rectangular world. In fact, your world is so bland that it could be represented by a $r \times c$ grid. Each cell on the grid represents a country. Each country has a dominant religion. There are only two religions in your world. One of the religions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

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 that 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 parameters x, d, s must be chosen in such a way that any of the missionary groups won't leave the grid.

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. Here, we've chosen a 1×4 subgrid, the direction NORTH, and $s = 2$ steps.



You are a being which believes in free will, for the most part. However, you just really want to stop receiving murders that 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

The first line of input contains a single integer t ($1 \leq t \leq 2 \cdot 10^4$) denoting 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 ($1 \leq r, c \leq 60$). The next r lines each contains c characters describing the dominant religions in the countries. In particular, the j-th character in the i-th 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.

It is guaranteed that the grid will only contain "A" or "P" characters. It is guaranteed that the sum of the $r \cdot c$ in a single file is at most $3 \cdot 10^6$.

Output

For each test case, output a single line containing the minimum number of usages of your power needed to convert everyone to Beingawesomeism, or the string "MORTAL" (without quotes) if it is impossible to do so.

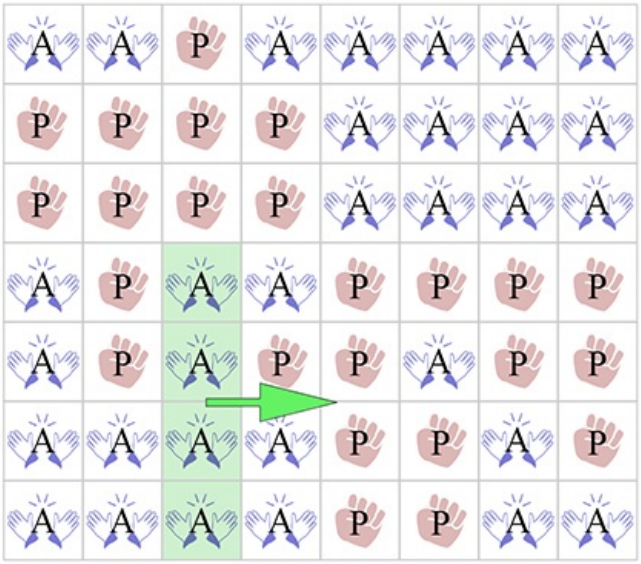
Example

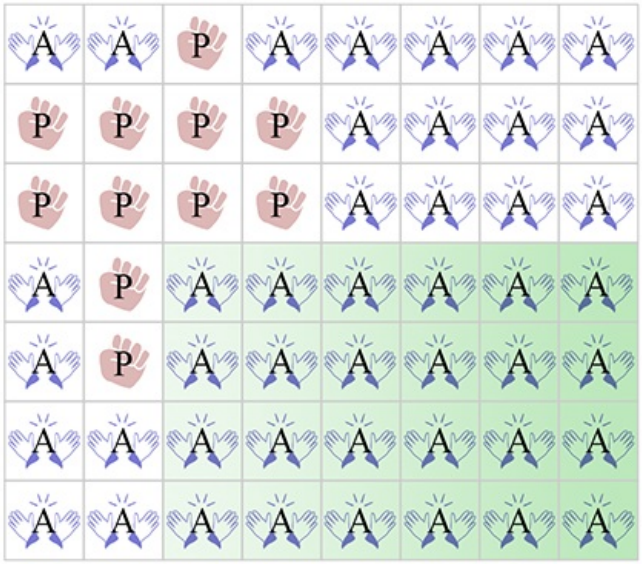
input
4 7 8 AAPAAAAA PPPPAAAA PPPPAAAA APAAPPPP APAPPAPP AAAAPPAP AAAAPPAA 6 5 AAAAA AAAAA AAPAA AAPAP AAAPP AAAPP 4 4 PPPP PPPP PPPP PPPP 3 4 PPPP PAAP PPPP
output
2 1 MORTAL 4

Note

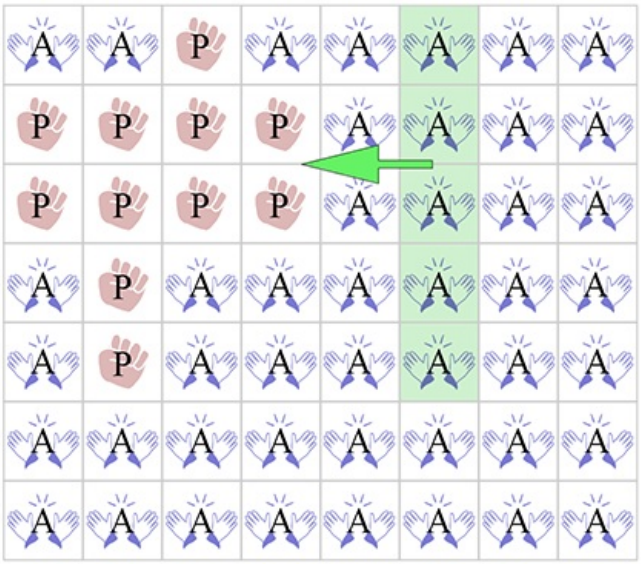
In the first test case, it can be done in two usages, as follows:

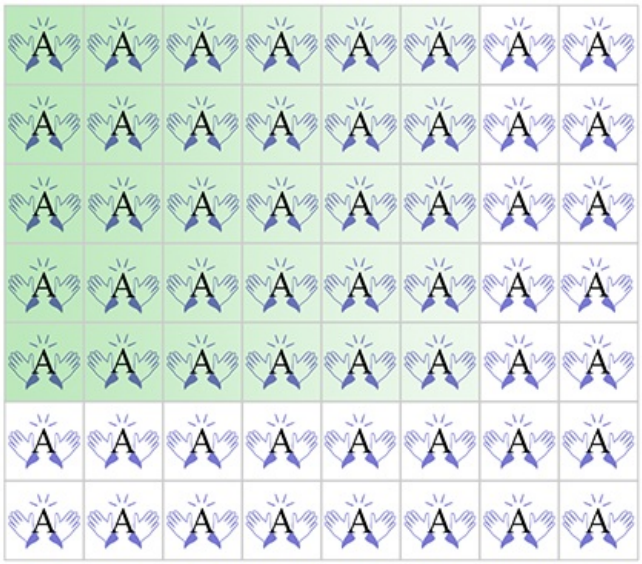
Usage 1:





Usage 2:





In the second test case, it can be done with just one usage of the power.

In the third test case, it is impossible to convert everyone to Beingawesomeism, so the answer is "MORTAL".

C. Jeremy Bearimy

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

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 that will either make people happier or torture them for eternity.

You have a list of k pairs of people who 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 exactly one house, and each house will have exactly 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. In other words, the graph with the houses as vertices and the roads as edges is a tree.

The truth is, these k pairs of people are actually soulmates. We index them from 1 to k. We denote by $f(i)$ the amount of time it takes for the i-th pair of soulmates 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 $f(i)$ over all pairs i is minimized. Let's define this minimized sum as 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 $f(i)$ over all pairs i is

maximized. Let's define this maximized sum as B. This makes sure that soulmates will have a difficult time to visit each other.

What are the values of G and B?

Input
The first line of input contains a single integer t ($1 \le t \le 500$) denoting the number of test cases. The next lines contain descriptions of the test cases.

The first line of each test case contains a single integer k denoting the number of pairs of people ($1 \le k \le 10^5$). The next $2k - 1$ lines describe the roads; the i-th of them contains three space-separated integers a_i, b_i, t_i which means that the i-th road connects the a_i-th and b_i-th houses with a road that takes t_i units of time to traverse ($1 \le a_i, b_i \le 2k$, $a_i \neq b_i$, $1 \le t_i \le 10^6$). It is guaranteed that the given roads define a tree structure.

It is guaranteed that the sum of the k in a single file is at most $3 \cdot 10^5$.

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

Example

input
2 3 1 2 3 3 2 4 2 4 3 4 5 6 5 6 5 2 1 2 1 1 3 2 1 4 3
output
15 33 6 6

Note
For the sample test case, we have a minimum sum equal to $G = 15$. One way this can be achieved is with the following assignment:

- The first pair of people get assigned to houses 5 and 6, giving us $f(1) = 5$;
- The second pair of people get assigned to houses 1 and 4, giving us $f(2) = 6$;
- The third pair of people get assigned to houses 3 and 2, giving us $f(3) = 4$.

Note that the sum of the $f(i)$ is $5 + 6 + 4 = 15$.

We also have a maximum sum equal to $B = 33$. One way this can be achieved is with the following assignment:

- The first pair of people get assigned to houses 1 and 4, giving us $f(1) = 6$;
- The second pair of people get assigned to houses 6 and 2, giving us $f(2) = 14$;
- The third pair of people get assigned to houses 3 and 5, giving us $f(3) = 13$.

Note that the sum of the $f(i)$ is $6 + 14 + 13 = 33$.

D. Miss Punyverse

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

The Oak has n nesting places, numbered with integers from 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 $\mathcal{P}$ is called a partition if each of the n nesting places is contained in exactly one of the villages in $\mathcal{P}$. In other words, no two villages in $\mathcal{P}$ 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 $\mathcal{P}$ 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 **strictly** 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 from 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
The first line of input contains a single integer t ($1 \leq t \leq 100$) denoting the number of test cases. The next lines contain descriptions of the test cases.

The first line of each test case contains two space-separated integers n and m ($1 \leq m \leq n \leq 3000$). The second line contains n space-separated integers $b_1, b_2, \ldots, b_n$ ($0 \leq b_i \leq 10^9$). The third line contains n space-separated integers $w_1, w_2, \ldots, w_n$ ($0 \leq w_i \leq 10^9$). The next $n-1$ lines describe the pairs of adjacent nesting places. In particular, the i-th of them contains two space-separated integers x_i and y_i ($1 \leq x_i, y_i \leq n$, $x_i \neq y_i$) denoting the numbers of two adjacent nesting places. It is guaranteed that these pairs form a tree.

It is guaranteed that the sum of n in a single file is at most 10^5.

Output
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.

Example

input
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
output
2 0

Note
In the first test case, we need to partition the $n = 4$ nesting places into $m = 3$ villages. We can make Ugly Wasp win in 2 villages via the following partition: $\{\{1, 2\}, \{3\}, \{4\}\}$. In this partition,

- Ugly Wasp wins in village $\{1, 2\}$, garnering 181 votes as opposed to Pretty Bee's 170;
- Ugly Wasp also wins in village $\{3\}$, garnering 111 votes as opposed to Pretty Bee's 70;
- Ugly Wasp loses in the village $\{4\}$, garnering 0 votes as opposed to Pretty Bee's 50.

Thus, Ugly Wasp wins in 2 villages, and it can be shown that this is the maximum possible number.

In the second test case, we need to partition the $n = 2$ nesting places into $m = 1$ village. There is only one way to do this: $\{\{1, 2\}\}$. In this partition's sole village, Ugly Wasp gets 563 votes, and Pretty Bee also gets 563 votes. Ugly Wasp needs strictly more votes in order to win. Therefore, Ugly Wasp doesn't win in any village.

E. Kirchhoff's Current Loss

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

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.

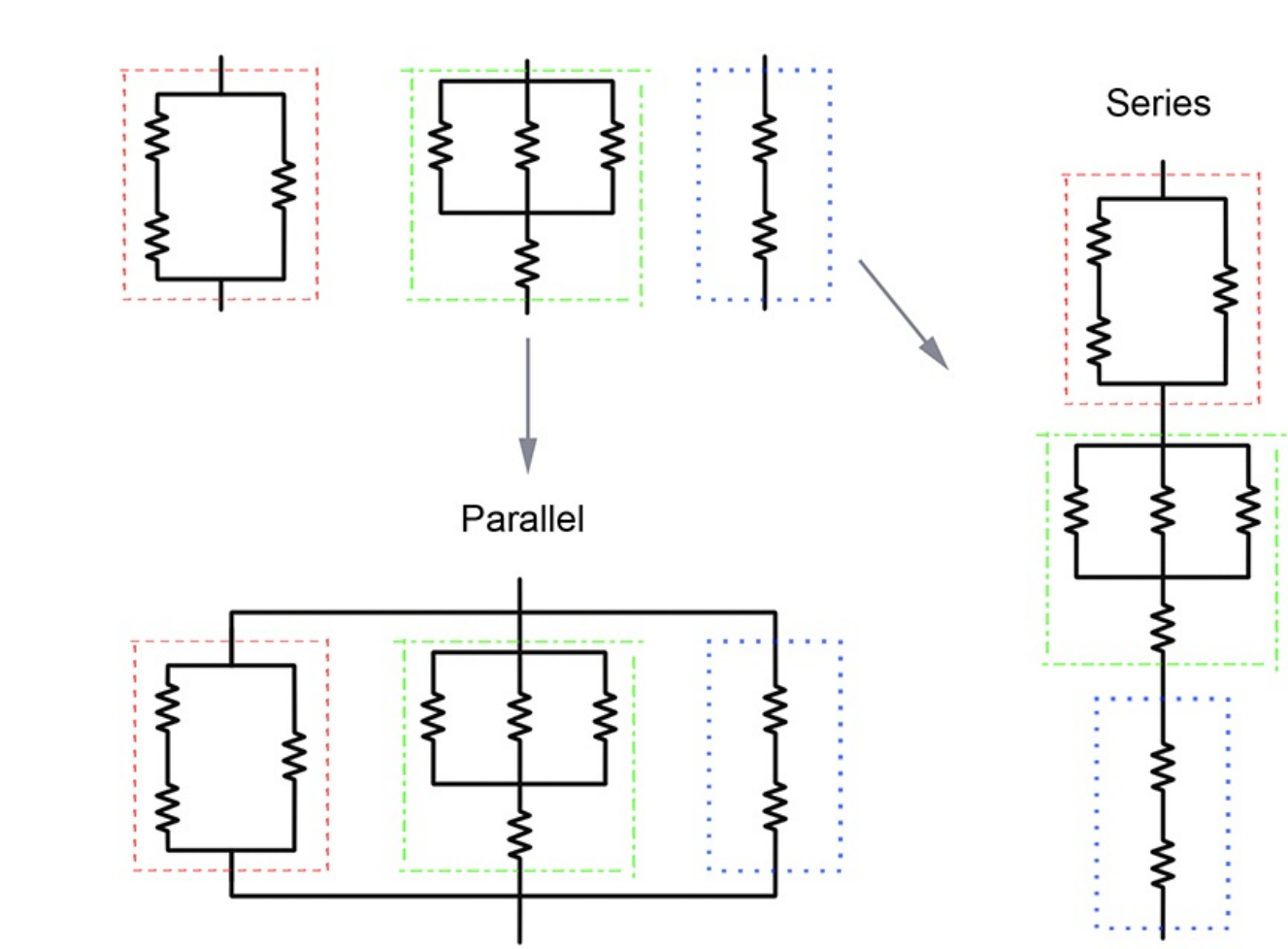
The negativity just keeps flowing out of him, but even after complaining so many times he still hasn't lepton the chance to directly change anything.

"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 while keeping the same circuit layout and maintaining the same effective resistance.

A circuit has two endpoints, and is associated with a certain constant, R, called its **effective resistance**.

The circuits we'll consider will be formed from individual resistors joined together in series or in parallel, forming more complex circuits. 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 this way:

- 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

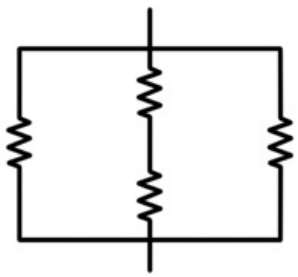
$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_k},$$

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

Circuits will be represented by strings. Individual resistors are represented by an asterisk, "*". For more complex circuits, suppose $s_1, s_2, \ldots, s_k$ represent $k \geq 2$ circuits. Then:

- "(`s1 S s2 S ... S sk`)" represents their series circuit;
- "(`s1 P s2 P ... P sk`)" represents their parallel circuit.

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



Given 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;
- The sum of the resistances of the individual resistors is minimized.

If there are n individual resistors, then you need to output the list $r_1, r_2, \ldots, r_n$ ($0 \le r_i$, and r_i is an integer), where r_i is the resistance assigned to the i-th individual resistor that appears in the input (from left to right). If it is impossible to accomplish the task, you must say so as well.

If it is possible, then it is guaranteed that the minimum sum of resistances is at most 10^{18}.

Input
The first line of input contains a single integer t ($1 \le t \le 32000$), denoting the number of test cases. The next lines contain descriptions of the test cases.

Each test case consists of a single line containing the integer r ($1 \le r \le 10^6$), space, and then the string representing the circuit. It is guaranteed that the string is valid and follows the description above. The number of individual resistors (symbols "*") is at least 1 and at most 80000.

It is guaranteed that the total number of individual resistors across all test cases is at most 320000.

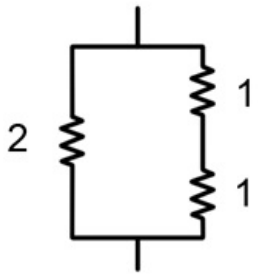
Output
For each test case, print a single line:

- If it's possible to achieve an effective resistance of r, print "REVOLTING" (without quotes) and then n integers $r_1, r_2, \ldots, r_n$ — the resistances assigned to the individual resistors. Here, n denotes the number of the individual resistors in the circuit. There may be multiple possible such assignments with a minimal sum of resistances of the individual resistors, you can output any of them;
- If it's impossible, print the string: "LOSS" (without quotes).

Example

input
3 5 * 1 (* S *) 1 (* P (* S *))
output
REVOLTING 5 REVOLTING 1 0 REVOLTING 2 1 1

Note
The following illustrates the third sample case:



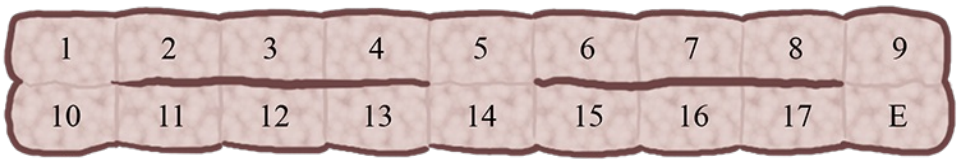
Here, the sum of the resistances of the individual resistors is $2 + 1 + 1 = 4$, which can be shown to be the minimum. Note that there may be other assignments that achieve this minimum.

F. Intergalactic Sliding Puzzle

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

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.

Input

The first line of input contains a single integer t ($1 \leq t \leq 4$) denoting the number of test cases. The next lines contain descriptions of the test cases.

Each test case consists of three lines. The first line contains a single integer k ($1 \leq k \leq 15$) which determines the size of the grid. Then two lines follow. Each of them contains $2k + 1$ space-separated integers or the letter E. They describe the first and second rows of organs, respectively. It is guaranteed that all $4k + 1$ organs are present and there is exactly one E.

Output

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 (possibly empty) 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;
- There must be no cycles involving the shortcuts that are reachable from the main sequence;
- The resulting sequence of moves is finite, after expanding all 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, then 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.

Example

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

output

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

Note

There are three shortcuts defined in the first sample output:

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

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

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