Code Jam to I/O for Women 2018 - Code Jam to I/O for Women

Centrists

Problem

Exactly three candidates are running for office. Each candidate's name is a single string of length **L** that is made up only of uppercase letters of the English alphabet, and no two candidates have the same name.

This area has a law to ensure that candidates will not consistently be at an advantage or disadvantage based on their names. Before each election, an ordering of the English alphabet will be chosen from among all possible orderings, and that ordering will be used to alphabetize the names before listing them on the ballot. (To determine which of two different names comes earlier in the list on the ballot, start by comparing the first letter of each name. If they are different, then the name with the first letter that comes earlier in the chosen ordering is the one that comes earlier in the list on the ballot. If they are the same, move on to comparing the second letter of each name, and so on.)

Each of the three candidates wants to adopt a "middle-of-the-road" image, and so they think that being in the middle of the list of three names on the ballot will be advantageous. For each candidate, determine whether there is at least one ordering of the English alphabet for which their name would be the second of the three names on the ballot. Note that we are considering each candidate independently.

For example, suppose that our candidates are named `BCB`, `CAB`, and `CBC`. Since the letters `D` through `Z` are not used in these names, we will only consider the relative ordering of the letters `A`, `B`, and `C`. If the ordering `A`, `B`, `C` is chosen, for example, then the candidates will be listed in the order `BCB`, `CAB`, `CBC`; this demonstrates that it is possible for `CAB` to be in the middle. If the ordering `A`, `C`, `B` is chosen, then the candidates will be listed in the order `CAB`, `CBC`, `BCB`; this demonstrates that it is possible for `CBC` to be in the middle. However, even under any of the other four possible orderings, it turns out that `BCB` can never be in the middle.

Input

The first line of the input gives the number of test cases, **T**; **T** test cases follow. Each begins with one line containing one integer **L**: the length of each candidate's name. Then, there is one line with three different strings $\mathbf{N_i}$ of uppercase letters of the English alphabet; the i-th of these is the name of the i-th candidate.

Output

For each test case, output one line containing `Case #x: y1 y2 y3`, where `x` is the test case number (starting from 1) and each `yi` is `YES` if it is possible for the i-th candidate to be in the middle, as described in the problem statement, and `NO` otherwise.

Limits

$1 \leq \mathbf{T} \leq 100$.
Time limit: 20 seconds per test set.
Memory limit: 1GB.
$1 \leq \mathbf{L} \leq 100$.

$\mathbf{N_i}$ is of length **L**, for all i.
$\mathbf{N_i} \neq \mathbf{N_j}$, for all i ≠ j.

Small dataset (Test set 1 - Visible)

$\mathbf{N_i}$ contains only uppercase letters from the set {`A`, `B`, `C`}, for all i.

Large dataset (Test set 2 - Hidden)

$\mathbf{N_i}$ contains only uppercase letters from the English alphabet, for all i.

Sample

Input	Output
3 3 BCB CAB CBC 2 CC CA AA 6 MEDIAN MEDIAL MEDIAS	Case #1: NO YES YES Case #2: NO YES NO Case #3: YES YES YES

Note that the last sample case would not appear in the Small dataset.

Sample Case #1 is the one described in the problem statement.

In Sample Case #2, no matter which of the two possible relative orderings of `A` and `C` is chosen, `CA` will be in the middle.

In Sample Case #3, any of the names can end up in the middle, depending on which of `L`, `N`, and `S` comes second in the relative order of those three letters in the ordering.