The Byzantine Generals
Problem ID: byzantine

You must implement the OM(m) algorithm described in *The Byzantine Generals Problem*. Your program will work by reading the specification of a Byzantine Generals problem from standard input, and writing a table with the orders received by the generals (the exact format is described below).

We will assume the following:

- The inputs to the algorithm are the following:
  - m: As defined in *The Byzantine Generals Problem*. Note that m doesn't denote the number of traitors; it denotes the level of recursion in the algorithm (which may not necessarily be enough to cope with the number of traitors, which is specified separately). You may assume that $m > 0$.
  - G: A list of n generals: $G_0, G_1, G_2, \ldots, G_n$. A general is either loyal or a traitor. G_0 is the commander general. The remaining $n - 1$ generals are the lieutenants.
  - o_C: The order the commander gives to its lieutenants ($o_C \in \{\texttt{ATTACK}, \texttt{RETREAT}\}$)
- If a general is loyal, and it has to relay an order o to general G_i, it always relays the value of o.
- If a general is a traitor, and it has to relay an order o to general G_i, it will relay the value of o if i is odd, and it will send the opposite of o if i is even. Note that, in the case of a traitorous commander general, the order being relayed is o_C.
- When performing a majority vote, only `ATTACK` and `RETREAT` votes are taken into account, and it is enough for one of the two to have a *relative* majority (i.e., a plurality) to determine what action wins the vote. If the number of `ATTACK` and `RETREAT` votes is the same, then the result of the vote is a tie.

Input

The input to your program will be a single line containing the specification of an instance of the Byzantine Generals problem. The line is composed of three values, each separated by a single space:

$$m \quad G_0G_1G_2 \ldots G_n \quad o_C$$

Where:

- m is a non-negative integer.
- $G_0G_1G_2 \ldots G_n$ is a string where each character represents whether G_i is loyal or a traitor. More specifically, the i^{th} character will be `L` if G_i is loyal, and `T` if G_i is a traitor.
- o_C is either `ATTACK` or `RETREAT`.

For example:

```
1 LLLT ATTACK
```

The above line specifies that you must run OM(1) with four generals (a loyal commander general, two loyal lieutenant generals, and a traitorous lieutenant general), and that the commander general's order is to attack.

Output

Your output will be $n - 1$ lines, each specifying the final state of each lieutenant general. Each line has three values, each separated by a single space:

$$o_{0,i} \quad o_{1,i}o_{2,i}o_{3,i} \ldots o_{n,i} \quad o'$$

Where:

- $o_{0,i}$ is the order received by G_i in OM(m) (i.e., in the very first round of message-passing).
- $o_{j,i}$ is the order that G_i will use from G_j for the purposes of doing a majority vote. Take into account that this does *not* necessarily correspond to the order sent from G_j to G_i (except when running the algorithm with $m = 1$). The value of $o_{j,i}$ may itself be the result of a majority vote done in further rounds of message-passing.
- o' is the final order that G_i will carry out based on the majority vote of the $o_{j,i}$ orders. This value can be `ATTACK`, `RETREAT`, or `TIE`. If the general is a traitor, add an asterisk ($\star$) after the order.
- All the $o_{j,i}$ (including $o_{0,i}$) are written with a single character:
  - `A`: Attack.
  - `R`: Retreat.
  - $-$: Tie.
  - When $j = i$, a single space is used (since a general does not receive messages from itself).

Sample Input 1

```
1 LLLT ATTACK
```

Sample Output 1

```
A  AA ATTACK
A A R ATTACK
A AA  ATTACK*
```

Sample Input 2

```
1 TLLL ATTACK
```

Sample Output 2

```
A  RA ATTACK
R A A ATTACK
A AR  ATTACK
```

Sample Input 3

```
1 TLLLLLT ATTACK
```

Sample Output 3

```
A  RARAR TIE
R A ARAA ATTACK
A AR RAR TIE
R ARA AA ATTACK
A ARAR R TIE
R ARARA  TIE*
```

Sample Input 4

```
2 TLLLLLT ATTACK
```

Sample Output 4

```
A  RARAR TIE
R A ARAR TIE
A AR RAR TIE
R ARA AR TIE
A ARAR R TIE
R ARARA  TIE*
```