Antiparticle Antiphysics

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

In an alternate universe, where the laws of physics are out of whack...

A new research facility has just been built. It is called the Large Antihadron Collider (LAC), the largest antiparticle collider of its kind, and antiphysicists are eager to use it to study something called "regular matter", which is similar to antimatter except with reversed charge, parity, and time.

In one of their LAC experiments, the antiphysicists successfully confined two kinds of particles, *antiprotons* and *protons*, in a container, where particles are lined up from left to right. We can represent the container's *state* as a 1-indexed string. The length of the string equals the number of particles in the container, and the i-th character of the string is `A` if the i-th particle from the left is an antiproton, or `P` if it is a proton.

Using the LAC's bizarro-energy beams, they can modify the state using any of four different types of operations:

Operation 1: Choose a particular proton, and then insert two antiprotons, one to its left and the other to its right. This has the effect of replacing the corresponding character `P` in the state string with `APA`.

Operation 2: Choose a particular antiproton, and then insert two protons, one to its left and the other to its right. This has the effect of replacing the corresponding character `A` in the state string with `PAP`.

Operation 3: Choose a contiguous subsequence of a antiprotons, and then remove them.

Operation 4: Choose a contiguous subsequence of p protons, and then remove them.

Note that the integers a in Operation 3 and p in Operation 4 are given in the input and are fixed.

These operations can be performed an arbitrary number of times in an arbitrary order, but only one operation can be performed at a time.

The *initial state* is represented by the string S. They would like to transform it into the *goal state* represented by the string E using a sequence of operations. Determine whether it is possible to do so. If it is possible, find one sequence of operations that transforms the initial state into the goal state.

Input

The first line of input contains one integer t ($1 \le t \le 10$) representing the number of test cases. After that, t test cases follow. Each of them consists of a single line containing two integers a and p ($5 \le a, p \le 20$) and two strings S and E ($1 \le |S|, |E| \le 50$, $S \ne E$). The strings S and E consist only of characters `A` and `P`.

Output

For each test case, output the following.

If the transformation is impossible, output `-1` in a single line.

Otherwise, on the first line, output an integer k representing the number of operations to transform the initial state into the goal state. On each of the next k lines, output one of the following, without the quotes, to describe one operation:

1. "`+P` i" to apply Operation 1 on the i-th particle from the left ($i \ge 1$). This particle must be a proton.

2. "`+A` i" to apply Operation 2 on the i-th particle from the left ($i \geq 1$). This particle must be an antiproton.

3. "`-A` i" to apply Operation 3 on the a consecutive particles whose leftmost particle is the i-th particle from the left ($i \geq 1$). These particles must be antiprotons.

4. "`-P` i" to apply Operation 4 on the p consecutive particles whose leftmost particle is the i-th particle from the left ($i \geq 1$). These particles must be protons.

These operations are performed in the order of the output lines and must transform the initial state into the goal state.

The number of operations k must satisfy $1 \leq k \leq 35\,000$. It can be shown that there's always a sequence of operations satisfying this bound for k if the initial state can be transformed into the goal state at all. Any valid sequence satisfying this bound for k will be accepted. In particular, you do not need to minimize the value of k.

Example

standard input	standard output
4 13 10 PP PAAAAPAAAA 10 13 AAAAAAA PPPPPPP 7 8 PPAAAAAAAAP PPAP 8 9 PAPPPPPPPPP PPAP	4 +P 2 +P 3 +P 4 +P 5 -1 1 -A 3 2 +A 2 -P 5

Note

Explanation for the sample input/output #1

In the first test case, the sequence of state string operations is
`PP` $\rightarrow$ `PAPA` $\rightarrow$ `PAAPAA` $\rightarrow$ `PAAAPAAA` $\rightarrow$ `PAAAAPAAAA`.

In the fourth test case, the sequence of state string operations is
`PAPPPPPPPPP` $\rightarrow$ `PPAPPPPPPPPPP`, then `PPAPPPPPPPPPP` $\rightarrow$ `PPAP`.