292 Presentation Error

One of the main burdens of the Jury of the Scholastic Programming Contest is not to decide whether a submitted program is incorrect, but how to classify the error. In the past, we had `´Failed Testcase´`, `´Wrong Answer´`, `´Wrong Output Format´` and `´Too much/Too Little Output´` to worry about.

The interpretation of these messages depended largely on the jury member involved. For instance, while some believe that `´Wrong Answer´` indicates that all answers are wrong, and `´Failed Testcase´` applies when at least one answer is right, others feel that `´Wrong Answer´` should be used if more than one answer is wrong, and `´Failed Testcase´` only if exactly one test went wrong.

Fortunately, all these worries are gone, since now we only need to distinguish between `´Wrong Answer´`, `´Presentation Error´` and `´Accepted´` (all other messages are the result of compilation errors, run-time errors, and non-terminating programs).

To eliminate any subjectivity in deciding between a `Presentation Error` and a `Wrong Answer`, the Jury of this year's Programming Contest has defined an exact procedure to determine whether a program produces a `´Wrong Answer´`, a `´Presentation Error´`, or should be `Accepted`.

In the description of the rules, we distinguish between JuryOut and SubmitOut, as the output intended by the Jury, and the output submitted, respectively. The JuryOut contains parts which are considered essential in the output of a correct algorithm. Those essentials are placed between '[' and ']'. Those brackets are not part of the output, thus they should not appear in SubmitOut. The algorithm to decide between `´Accepted´`, `´Wrong Answer´` and `´Presentation Error´` is as follows:

1. From each line in both outputs, all trailing white space (blanks and tabs) should be removed. After that, all trailing empty lines should be removed.
2. If after step 1, JuryOut and SubmitOut are identical, the algorithm returns `´Accepted´`.
3. All letters in both outputs are changed to uppercase (including those between square brackets).
4. We name the essentials E_1 through E_n.
5. If each of the strings E_1 through E_n occurs as a string in SubmitOut, and E_i comes after (without overlapping) E_{i-1}, for all $2 \leq i \leq n$, then the algorithm returns `´Presentation Error´`.
6. The algorithm returns `´Wrong Answer´`.

As we (the Jury) need to have such a program (and we need it NOW), your job is to write it for us.

Input Specification

The input contains on the first line the number of test cases (N). Each test case has on its first line the number (J) of lines in JuryOut, and the number (S) of SubmitOut lines, separated by a single space. Then follow the J lines of JuryOut and the S lines of SubmitOut. Both JuryOut and SubmitOut are no longer than 10 lines. A line is at most 80 characters long. Essentials are non-empty strings that do not cross line boundaries. The first and last characters of an essential are not white space. Essentials will not be nested.

Output Specification

The output has to be `´Accepted´`, `´Wrong Answer´`, or `´Presentation Error´` on a single line for each test case.

Example Input

```
4
1 2
Just one line?
Just one line?

2 2
The first characters of the alphabet are:
[abcde]
Here they come:
a b c d e
1 1
That´s it: [abcde]
That´s it: AbCdE
1 1
[2] and [3] make [5]
I guess 2 and 3 are less than 50.
```

Example Output

```
Accepted
Wrong Answer
Presentation Error
Presentation Error
```