UVa Online Judge

831 Document Validator

Nowadays mark–up languages like `LaTeX`, `HTML`, `SGML`, and `XML` are widely used to define, in a completely textual mode, the structure of documents. Those languages are composed by tags, or annotations, that are used to mark blocks of text (their begin and end) in order:

- to specify the document structure – for instance, the *front material* like the *title*, the *authors*, the *date*, or a *chapter*, with its *sections* and *paragraphs*;
- to give them some special interpretation, or a particular formatting information – for instance, to say that the block designates a *country*, a *profession*, a *king*, or the block must be printed out in *italic*, or *boldface*.

Tags are just words like all the others that belong to the original plain text; so, it is necessary to use some lexical convention to distinguish those special words (annotations). Also, we need some convention to identify the beginning and the end of the text block we want to annotate.

In the present context, our mark–up language follows an approach similar to the `SGML` based languages:

- special characters – square `[ ]` and curly `{ }` brackets – are used to identify the words that are tag names;
- the same tag name is used to setup block boundaries, and a lexical detail defines the opening and closing tag. The convention is: `[opening-tag[, ]closing-tag]` and `{opening-closing-tag}`.

Paired tags are employed to define the document structure, while single tags are used to give meaning or formatting information. Tag names (one letter followed by zero or more letters or digits) are free, i.e. not previously defined; this means that the user of that mark–up language can choose the names of the tags he will employ to annotate his document. However, to be a **valid annotation** it must be in conformity with the following rules:

- an *opening–tag* must always have a corresponding *closing–tag*;
- a *closing–tag* must always have a corresponding *opening–tag*;
- paired tags may be nested to any level, however the last opening–tag must be closed before any enclosing tag, i.e. one external block can not be closed before an internal one;
- an *opening–closing–tag* can not have other tags inside; in that case, the corresponding block of text will appear inside the curly brackets.

Given a structured document, supposed to be annotated accordingly to the mark–up language above described, write a program that validates it, that is that verifies if the tags are properly used. If the document is valid (has no errors), your program must identify all the different tag names used, separating structural and semantic ones.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The input is a plain text file with tags, following the above stated conventions. Assume that lexical conventions specified for the three tag types (`[opening-tag[`, `]closing-tag]` and `{opening-closing-tag}`) are always complied.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The output consists of:

- just 1 line with the word '`ERROR`' if one of the rules above is not observed, making the input an invalid annotated document;
- 2 lists with the tag names found, without duplicates and in the order they appear in the document (from the beginning to the end), 1 word per line. The first list begins with the heading line '`STRUCTURAL TAGS`', while the second list will begin with the heading line '`SEMANTIC TAGS`'. Blank lines between those are not allowed.

Sample Input

```
1

[memo[
[de[ Comiss ao Cient fica do MIUP ]de]
[para[ Todos os Concorrentes ]para]
[data[{bold 2001.set.25}]data]
[mensagem[
[parag[Devem ter o m aximo cuidado na leitura dos enunciados.]parag]
[parag[Desejamos a todos {desejo Calma} e {desejo Boa Sorte}!]parag]
]mensagem]
]memo]
```

Sample Output

```
STRUCTURAL TAGS
memo
de
para
data
mensagem
parag
SEMANTIC TAGS
bold
desejo
```