Any fool can do it

Surely you know someone who thinks he is very clever. You decide to let him down with the following problem:

- "Can you tell me what the syntax for a set is?", you ask him.
- "Sure!", he replies, "a set encloses a possibly empty list of elements within two curly braces. Each element is either another set or a letter of the given alphabet. Elements in a list are separated by a comma."
- "So if I give you a word, can you tell me if it is a syntactically correct representation of a set?"
- "Of course, any fool can do it!" is his answer.

Now you got him! You present him with the following grammar, defining formally the syntax for a set (which was described informally by him):

```
Set          ::= "{" Elementlist "}"
Elementlist  ::= <empty> | List
List         ::= Element | Element "," List
Element      ::= Atom | Set
Atom         ::= "{" | "}" | ","
```

<empty> stands for the empty word, i.e. the list in a set can be empty.
Soon he realizes that this task is much harder than he has thought, since the alphabet consists of the characters which are also used for the syntax of the set. So he claims that it is not possible to decide efficiently if a word consisting of "{", "}" and "," is a syntactically correct representation of a set or not.
To disprove him, you need to write an efficient program that will decide this problem.

Input Specification

The first line of the input file contains a number representing the number of lines to follow.
Each line consists of a word, for which your program has to decide if it is a syntactically correct representation of a set. You may assume that each word consists of between *1* and *200* characters from the set { "{", "}", "," }.

Output Specification

Output for each test case whether the word is a set or not. Adhere to the format shown in the sample output.

Sample Input

```
4
{}
{{}}
{{}},{,}}
{,,}
```

Sample Output

Word #1: Set
Word #2: Set
Word #3: Set
Word #4: No Set