3. CLUNK is a simple programming language that contains only assignments and conditional branches. A CLUNK program has the form

`0:` S_0`; 1:` S_1`; ⋯` N`: stop;`

The consecutive decimal numerals before the colons are *statement labels*; there is no upper bound on N. Input is in free form; there may be arbitrary whitespace around the statement labels, the components of the statements (the S_i) and the punctuation marks, but not within a component (e.g., `stop` must appear as shown, with no spaces separating its letters). Each statement ends with a semicolon.

Only the last statement in a CLUNK program is `stop`. Each of the other S_i has either the form of

- an *assignment:*

  I `= F(`$E_1, E_2, \cdots, E_k$`)`

  —where $k \geq 0$ and I and all of the E_i are *variables*, written as letters (upper- or lower-case, with 'a' and 'A' being different); or

- a *conditional branch:*

  `if F(`$E_1, E_2, \cdots, E_k$`) goto` M

  where the E_i and k are as before and M is a statement label (between 0 and N, inclusive).

The value k can be different in each statement. When $k = 0$, the expression is `F()`. (The `F` is just there for show; that's why it's the same all the time.)

You are to write a program to detect *use before definition* in CLUNK programs. That is, given any CLUNK program, you are to determine if it is possible that a variable in the program might apparently be *used* before it has been assigned a value (in compiler terminology, before it is *defined*). More specifically, you should report that use before definition is possible if there is some apparently possible sequence of statements from the input program that leads to a statement in which one of the E_i is a variable that does not appear to the left of any preceding assignment statement in the sequence. A sequence of CLUNK statements is *apparently possible* if

- Statement 0 is the first statement in the sequence, and
- If an assignment statement numbered j is in the sequence, statement $j + 1$ is the next statement in the sequence, and
- If a conditional branch statement numbered j and containing the clause `goto` L is in the sequence, then the next statement in the sequence is either number $j + 1$ or L.

For example, in the CLUNK program on the left, statement 3 might involve a use before definition (of the variable x).

```
0: y = F();
1: if F(y) goto 3;
2: x = F(y);
3: x = F(x, y);
4: stop;
```

```
0: y = F();
1: if F(y) goto 5;
2: x = F(y);
3: if F(y) goto 1;
4: if F() goto 6;
5: x = F(y);
6: x = F(x,y);
7: stop;
```

Even though `x` is set in statement 2, the sequence $< 0, 1, 3 >$, which does not contain statement 2, is possible according to the rules. The program on the right has no possible uses before definition; all apparently possible paths to statement 6 define `x`.

The input to your program will consist of zero or more programs in the specified format. For each input program, the output is to identify the input set, and then print either

```
No use before definition
```

or

```
Possible use before definition
```

as shown in the sample outputs below.

Input	Output

`0: y = F(); 1: if F(y) goto 3;`
`2: x = F(y);`
`3: x =`
`F(x, y);`
`4:`
`stop;`
 	
`0: y = F();`
`1: if F(y) goto 5;`
`2: x = F(y);`
`3: if F(y) goto 1;`
`4: if F() goto 6;`
`5: x = F(y);`
`6: x = F(x,y);`
`7: stop;`
	`Set 0: Possible use before definition`
	`Set 1: No use before definition`