566 Adam's Genes

You've been hired by Gemini Labs, the world leader in human cloning, to write decision support software. At Gemini, all clones are derived from "ADAM", a genetically perfect human. Something about his DNA makes him much easier to clone than normal humans. Not all the clones of Adam are the same, though, because geneticists introduce mutations, in the form of recessive genes, to learn more about genetics. For example, Bob might be a clone of Adam with a recessive gene for baldness added. Scientists would study Bob to see what subtle effects the gene may have. Clones of Bob would carry the recessive gene, as would clones of those clones, and so on. All is well as long as no clone derived from Bob is given a second recessive baldness gene. If that were to happen, a bald clone would be produced and the Cloning Board would shut Gemini down.

The software you are to write takes cloning requests from the research staff and evaluates them for consistency and safety. A collection of requests is "inconsistent" if it includes a clone that is not descended from Adam. A collection of requests is "unsafe" if it produces a clone with two identical recessive genes.

Input

Your program consumes a file of cloning requests, one per line. Here is the format of a cloning request:

< *request* > = clone_< *name* >_from_< *name* >< *genelist* >
< *genelist* > = NULL | _mutating_< *gene* >< *genes* >
< *genes* > = NULL | _< *gene* >< *genes* >
< *gene* > = 3 upper case alphabetical characters
< *name* > = 1 to 10 upper case alphabetical characters
_ = one blank

A typical cloning request is

```
clone BOB from ADAM mutating BLD HEM
```

Note: there is always exactly one space between words; the last character on a line is immediately followed by EOLN. There can be zero to ten mutations in a request. If there are no mutations in the request, the keyword "mutating" does not appear, e.g.,

```
clone BOB from ADAM
```

The input is guaranteed to satisfy the syntactic format specifications, and it is guaranteed to contain at most one cloning request per clone, i.e., "`clone BOB`" will appear no more than once as the beginning of an input line. Furthermore, you are to process requests as though only those definitions which precede it are in effect. Therefore, if you have the following input segment

```
clone BOB from ADAM
clone MIKE from TIM
clone TIM from BOB
```

your output would include

```
clone MIKE from TIM has no connection to ADAM
```

because at the time MIKE was cloned, there was no connection to ADAM. If an clone is not consistent and safe, then all subsequent clones from that clone should be reported as having no connection to ADAM. For example, if you have the following input segment

```
clone BOB from ADAM mutating BLD
clone CHARLIE from BOB mutating BLD
clone DAVID from CHARLIE
```

your output would include

```
clone BOB from ADAM is consistent and safe
clone CHARLIE from BOB was at least twice mutated with BLD
clone DAVID from CHARLIE has no connection to ADAM
```

You are also guaranteed that no gene is listed twice in the same request.

Output

Your program produces a file of the processed requests, one per line, in the same order as they were consumed. The requests are modified according to the following rules.

1. If a clone is consistent and safe, the line should have the format

   `clone JOE from ADAM is consistent and safe`

2. If a clone is inconsistent, the line should indicate this as follows

   `clone` < *name* > `from` < *name* > `has no connection to ADAM`

3. If a clone is unsafe, the line should indicate this as follows

   `clone` < *name* > `from` < *name* > `was at least twice mutated with` < *gene* >

   where < *gene* > is the first gene to appear in the clone's mutation list that is a second mutation from Adam. You should print ONLY the first such doubly mutated gene.

If a particular cloning request is inconsistent, there is no need to report whether or not it is safe. Your output should contain exactly one space between words and no leading or trailing spaces.

Sample Input

```
clone JOE from ADAM
clone BOB from ADAM mutating HEM
clone SAM from BOB mutating BLD
clone ED from SAM mutating BLD
clone FRANK from ED mutating HEM
clone KAIN from ABEL
clone ABEL from KAIN
```

Sample Output

```
clone JOE from ADAM is consistent and safe
clone BOB from ADAM is consistent and safe
clone SAM from BOB is consistent and safe
clone ED from SAM was at least twice mutated with BLD
clone FRANK from ED has no connection to ADAM
clone KAIN from ABEL has no connection to ADAM
clone ABEL from KAIN has no connection to ADAM
```