Problem 5—Kissin' Cousins

Most people are unaware of the etymological origin of the phrase "kissing cousin." It is a variation of "kissing kin," itself a corruption of "kith and kin." The phrase refers to any member of one's extended family, and does not necessarily refer to Kentucky marriage practices.

Any two persons whose closest common ancestor is $(m+1)$ generations away from one person and $(m+1)+n$ generations from the other are mth. cousins n times removed. Normally $m \geq 1$ and $n \geq 0$, but for this problem, we allow m to be zero. A zeroth cousin zero times removed refers to brothers and sisters. Zeroth cousins once removed represents the relationship between an uncle/aunt and a niece/nephew. If two people are mth. cousins n times removed, we write this relationship as "cousin-m-n." If one person is an ancestor of the other, p generations away, we write their relationship as "descendant-p." For example, the relationship between grandparent and grandchild is descendant-2.

A relationship cousin-m_1-n_1 is closer than a relationship cousin-m_2-n_2 if either $m_1 < m_2$ or both $m_1=m_2$ and $n_1 < n_2$ are true. A relationship descendant-p_1 is closer than a relationship descendant-p_2 if $p_1 < p_2$. A descendant relationship is always closer than a cousin relationship.

Write a program that accepts definitions of simple relationships between individuals and displays the closest cousin or descendant relationship, if any, which exists between arbitrary pairs of individuals.

Notes: For this problem, assume that any parent can have any number of children, and that any child can have any number of parents (in sharp contradiction to biology).

You may assume that no circular relationship will be entered; for example, nobody will be made to be his own grandfather.

If one person is entered as being the distant ancestor of another person, don't assume that this ancestral line contains any other ancestors previously entered. Instead assume all the intermediate generations consist of unentered people. (Refer to the first note.)

INPUT SPECIFICATION

Each line in the input begins with #, R, F, or E. No line will exceed 80 characters in length, discarding the **<EOLN>**.

The last line in the output begins with E. Nothing will appear after the line except for **<EOLN>**. This line is not to be processed; it exists simply to signify the end of input data. This is the only line in the file that will begin with E.

lines are comments. Ignore everything on this line up to and including **<EOLN>**.

R lines direct your program to record a relationship between two different individuals. The first 5 characters following the R constitute the name of the first person; the next 5 characters constitute the name of the second. Case is significant, and the name may contain spaces or any printable ASCII character. Following the names and possibly separated from them by one or more spaces is a positive integer, k, defining the relationship. If k is 1, then the first names person is a child of the seond. If k is 2, then the first named person is a grandchild of the second, and so forth. Anything following the integer up to and including the **<EOLN>** is a comment.

F lines are queries. As above, the first 5 characters following the F constitute the name of the first person; the next 5 characters constitute the name of the second; your program is to find the closest relationship, if any, which exists between these two different persons. Anything following these names up to and including the **<EOLN>** is a comment. A query should be answered only with regard

to R lines which precede the query in the input. You may assume that F lines will only contain names previously entered on R lines.

OUTPUT SPECIFICATION

The output lines should appear in the order in which their corresponding F lines appear in the input. If the first name on an F line is XXXXX and the second name is YYYYY, the corresponding output line should be one of

```
XXXXX and YYYYY are descendant-p.<EOLN>
XXXXX and YYYYY are cousin-m-n.<EOLN>
XXXXX and YYYYY are not related.<EOLN>
```

whichever is appropriate, with m, n, and p replaced by the appropriate integers as described above. Remember that the relationship to be printed should be the closest one.

SAMPLE INPUT

```
# A comment!<EOLN>
RFred Joe  1 Fred is Joe's son<EOLN>
RFran Fred 2<EOLN>
RJake Fred 1<EOLN>
RBill Joe  1<EOLN>
RBill Sue  1<EOLN>
RJean Sue  1<EOLN>
RJean Don  1<EOLN>
RPhil Jean 3<EOLN>
RStan Jean 1<EOLN>
RJohn Jean 1<EOLN>
RMary Don  1<EOLN>
RSusanMary 4<EOLN>
RPeg  Mary 2<EOLN>
FFred Joe  <EOLN>
FJean Jake <EOLN>
FPhil Bill <EOLN>
FPhil Susan<EOLN>
FJake Bill <EOLN>
FDon  Sue  <EOLN>
FStan John <EOLN>
FPeg  John <EOLN>
FJean Susan<EOLN>
FFran Peg  <EOLN>
FJohn Avram<EOLN>
RAvramStan  99<EOLN>
FJohn Avram<EOLN>
FAvramPhil <EOLN>
E<EOLN>
<EOF>
```

SAMPLE OUTPUT

```
Fred  and Joe   are descendant-1.<EOLN>
Jean  and Jake  are not related.<EOLN>
Phil  and Bill  are cousin-0-3.<EOLN>
Phil  and Susan are cousin-3-1.<EOLN>
Jake  and Bill  are cousin-0-1.<EOLN>
Don   and Sue   are not related.<EOLN>
Stan  and John  are cousin-0-0.<EOLN>
Peg   and John  are cousin-1-1.<EOLN>
Jean  and Susan are cousin-0-4.<EOLN>
Fran  and Peg   are not related.<EOLN>
John  and Avram are not related.<EOLN>
John  and Avram are cousin-0-99.<EOLN>
Avram and Phil  are cousin-2-97.<EOLN>
<EOF>
```