CHAIN INSPECTION

CHALLENGE DESCRIPTION:

We use special generator to produce chains of elements. Don't ask why we need them, we're not sure that somebody knows the answer for this question. A chain is represented by a string of name-address pairs. So the first element is the name of a pair and the second one (address) is pointing to the name of a next pair. E.g.

```
BEGIN-3;4-2;3-4;2-END # GOOD
77-END;BEGIN-8;8-11;11-77 # GOOD
```

In examples above we can pass trough the chains from the 'BEGIN' to the 'END' without missing any single pair. In the first case we moved from 'BEGIN' to 3, from 3 to 4, from 4 to 2, from 2 to 'END'. In the second case we moved from 'BEGIN' to 8, from 8 to 11, from 11 to 77, from 77 to 'END'.

Our generator was producing only good chains, but something went wrong and now it generates random chains and we are not sure if it's a good chain or a bad one. E.g.

```
BEGIN-3;4-3;3-4;2-END # BAD
77-END;BEGIN-8;8-77;11-11 # BAD
```

In the first case the 'END' is unreachable because we have a loop between 3 and 4.
In the second case we can reach the 'END' but we missed one pair (11-11).
We know that for a BAD chain the generator first produces a GOOD chain but after that it may replace existing address in some pairs with an address from another pair. It never replaces an address in a pair to an addresses which isn't present in original chain.
You can help us by writing a program that investigates the input and finds GOOD and BAD chains.

INPUT SAMPLE:

Your program should accept as its first argument a path to a filename. Each string in this file is a chain. The pairs are separating by semicolon, the names and the address are separating by dash. E.g.

```
4-2;BEGIN-3;3-4;2-END
4-2;BEGIN-3;3-4;2-3
```

OUTPUT SAMPLE:

For each line of input print out the chain status. E.g.

```
GOOD
BAD
```

Constraints:
The number of pairs in a chain is in range [1, 500]
The addresses are integers in range [2, 10000]