Source: Codeforces **Time Limit:** 1000ms **Memory Limit:** 512MB

Suffix Structures

Bizon the Champion isn't just a bison. He also is a favorite of the "Bizons" team.

At a competition the "Bizons" got the following problem: "You are given two distinct words (strings of English letters), s and t. You need to transform word s into word t".

The task looked simple to the guys because they know the suffix data structures well. Bizon Senior loves *suffix automaton*. By applying it once to a string, he can **remove** from this string any single character. Bizon Middle knows *suffix array* well. By applying it once to a string, he can **swap** any two characters of this string. The guys do not know anything about the suffix tree, but it can help them do much more.

Bizon the Champion wonders whether the "Bizons" can solve the problem. Perhaps, the solution do not require both data structures. Find out whether the guys can solve the problem and if they can, how do they do it? Can they solve it either only with use of *suffix automaton* or only with use of *suffix array* or they need both structures? Note that any structure may be used an unlimited number of times, the structures may be used in any order.

Input Format

- The first line contains a non-empty word s. The second line contains a non-empty word t.
- Words s and t are different. Each word consists only of lowercase English letters. Each word contains at most 100 letters.

Output Format

- In the single line print the answer to the problem:
  - Print "need tree" (without the quotes) if word s cannot be transformed into word even with use of both *suffix array* and *suffix automaton*.
  - Print "automaton" (without the quotes) if you need only the *suffix automaton* to solve the problem.
  - Print "array" (without the quotes) if you need only the *suffix array* to solve the problem.
  - Print "both" (without the quotes), if you need both data structures to solve the problem.
- It's guaranteed that if you can solve the problem only with use of *suffix array*, then it is impossible to solve it only with use of *suffix automaton*. This is also true for *suffix automaton*.

Sample test

input copy

```
automaton
tomat
```

output copy

```
automaton
```

input copy

```
array
arary
```

output copy

```
array
```

input copy

```
both
hot
```

output copy

```
both
```

input copy

```
need
tree
```

output copy

```
need tree
```

Explanation for sample test

In the third sample you can act like that: first transform "both" into "oth" by removing the first character using the *suffix automaton* and then make two swaps of the string using the *suffix array* and get "hot".