

# No Prefix Set

There is a given list of strings where each string contains only lowercase letters from  $a - j$ , inclusive. The set of strings is said to be a **GOOD SET** if no string is a prefix of another string. In this case, print **GOOD SET**. Otherwise, print **BAD SET** on the first line followed by the string being checked.

**Note** If two strings are identical, they are prefixes of each other.

### Example

*words* = ['abcd', 'bcd', 'abcde', 'bcde']

Here 'abcd' is a prefix of 'abcde' and 'bcd' is a prefix of 'bcde'. Since 'abcde' is tested first, print

```
BAD SET
abcde
```

*words* = ['ab', 'bc', 'cd'].

No string is a prefix of another so print

```
GOOD SET
```

### Function Description

Complete the *noPrefix* function in the editor below.

*noPrefix* has the following parameter(s):

- *string words[n]*: an array of strings

### Prints

- *string(s)*: either **GOOD SET** or **BAD SET** on one line followed by the word on the next line. No return value is expected.

### Input Format

First line contains *n*, the size of *words*[].

Then next *n* lines each contain a string, *words*[*i*].

### Constraints

$1 \leq n \leq 10^5$

$1 \leq \text{length of words}[i] \leq 60$

All letters in *words*[*i*] are in the range 'a' through 'j', inclusive.

### Sample Input00

STDIN	Function
-----	-----
7	words[] size n = 7
aab	words = ['aab', 'defgab', 'abcde', 'aabcde', 'bbbbbbbbbb', 'jabjjjad']
defgab	
abcde	
aabcde	

```
cedaaa  
bbbbbbbbbb  
jabjjjad
```

### Sample Output00

```
BAD SET  
aabcde
```

### Explanation

'aab' is prefix of 'aabcde' so it is a **BAD SET** and fails at string 'aabcde'.

### Sample Input01

```
4  
aab  
aac  
aacghgh  
aabghgh
```

### Sample Output01

```
BAD SET  
aacghgh
```

### Explanation

'aab' is a prefix of 'aabghgh', and 'aac' is prefix of 'aacghgh'. The set is a **BAD SET**. 'aacghgh' is tested before 'aabghgh', so and it fails at 'aacghgh'.