Mirko has an array of *N* **different** words that he wants to encrypt using a substitution cypher.

We encrypt the text using a substitution cypher by first choosing a *key* – a permutation of the English alphabet. Then we replace all occurrences of letter 'a' with the first letter of the key, all occurrences of letter 'b' with the second letter of the key, and so on until letter 'z'.

Besides the words, Mirko has an array *A* consisting of numbers from 1 to *N* given in a certain order (in other words, array *A* is a permutation of numbers from 1 to *N*). Mirko wants to pick a key such that the array of words after encrypting and lexicographic sorting corresponds to array *A*. More precisely, he wants the word initially located at A_i to be at location *i* after encryption and sorting.

Let's recall that the lexicographic word order is the order in which the words appear in the dictionary. If we are comparing two words, going from left to right, we search for the first position in both words where the letters differ and, based on that, we determine which word is lexicographically smaller. If word *X* is the beginning of the word Y, then word X is lexicographically smaller than word *Y*.

Mirko is currently not in the mood for encrypting, so he kindly asks you to do it for him.

INPUT

The first line of input contains the integer N ($2 \leq N \leq 100$).
Each of the following *N* lines contains a single word that consists of at most 100 lowercase letters of the English alphabet. The words will be mutually distinct.
The last line contains *N* integers – the elements of array *A*.

OUTPUT

In the case when a solution doesn't exist, output "NE".
Otherwise, output "DA" in the first line, and in the second line output a word consisting of 26 different letters of the English alphabet – the key for the substitution cipher.
If multiple solutions exist, output any.

SCORING

In test cases worth 30 points total, the words will consist of only the first 6 letters of the English alphabet.

SAMPLE TESTS

input	**input**	**input**
2 ab bc 2 1	3 abc bcd add 1 2 3	3 bbb ccc ddd 2 3 1
output	**output**	**output**
DA bacdefghijklmnopqrst uvwxyz	NE	DA adbcefghijklmnopqrst uvwxyz

Note: Outputs are split into multiple lines due to lack of horizontal space.

Clarification of the first test case:
After encrypting, the words become “ba”, “ac”, after lexicographic sorting, the array becomes “ac”, “ba”, which means the first word ended up in the second spot, and the second word in the first spot.

Clarification of the third test case:
After encrypting, the words become “ddd”, “bbb”, “ccc”, after lexicographic sorting, the array becomes “bbb”, “ccc”, “ddd”, which means the first word ended up in the third spot, the third word in the second spot, and the second word in the first spot.