Connecting words
1 second, 256MB

A nice children play proceeds as follows. One child start with a word. The next one should say a word that "follows" to the first word. Then, repeatedly, the next child should say a word that follows the previous one, and so on. In this game, every word is of length **L**. We say that word $\mathbf{w_2}$ follows word $\mathbf{w_1}$ if the prefix of length **L-1** of $\mathbf{w_2}$ is exactly the suffix of length **L-1** of $\mathbf{w_1}$.

Consider the case where **L=5**. For example, ELLOF follows HELLO, but GOODE does not follow FOODE. A successful sequence of words starting from HELLO and ending with OFFIC is shown below.

HELLO → ELLOF → LLOFF → LOFFI → OFFIC

Clearly not every sequence of characters forms a word. You are given a list of **N** possible words and you have to answer **T** questions, as pairs of words $\mathbf{w_1}$ and $\mathbf{w_2}$, if you can form a sequence of words starting from $\mathbf{w_1}$ and ending with $\mathbf{w_2}$.

Input
The first line of the input contains three integers **L N** and **T** (1<=**L**<=10; 1<=**N**<=1,000; 1<=**T**<=10)

The next **N** lines contain all possible words. More specifically, line 1+**i**, for 1<=**i**<=**N**, contains one word of length **L**. Each word consists of upper-case alphabets.

The next **T** lines contains **T** questions. Line 1+**N**+**j**, for 1<=**j**<=**T**, contains two words $\mathbf{w_1}$ and $\mathbf{w_2}$ each of length **L**. It is guaranteed that $\mathbf{w_1}$ and $\mathbf{w_2}$ are words from the list.

Output
Your program should output, for each question, a message **yes** if it is possible for form a sequence of words starting from $\mathbf{w_1}$ and ending with $\mathbf{w_2}$. Otherwise, it should output **no**.

HELLO → ELLOF → LLOFF → LOFFI → OFFIC

Example

Input	Output
5 10 4 LOFFI HELLO LLOFF OFFIC ELLOF GOODE FOODE OODEN GARDE ARDEN HELLO OFFIC GOODE FOODE FOODE GARDE GARDE ARDEN	yes no no yes

Notes:
To read all words, you can define an array as follows

```
char words[1000][12];
```

and then `cin` for each word; e.g.:

```
for(int i=0; i<n; i++) {
  cin >> words[i];
}
```

If you have problems reading inputs, please ask the TA. With this string representation, you can either directly manipulate strings or use standard string functions in C.