A. Shuffle Hashing

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has built his own web service. Being a modern web service it includes login feature. And that always implies password security problems.

Polycarp decided to store the hash of the password, generated by the following algorithm:

1. take the password p, consisting of lowercase Latin letters, and shuffle the letters randomly in it to obtain p' (p' can still be equal to p);
2. generate two random strings, consisting of lowercase Latin letters, s_1 and s_2 (any of these strings can be empty);
3. the resulting hash $h = s_1 + p' + s_2$, where addition is string concatenation.

For example, let the password $p =$ "abacaba". Then p' can be equal to "aabcaab". Random strings $s1 =$ "zyx" and $s2 =$ "kjh". Then $h =$ "zyxaabcaabkjh".

Note that no letters could be deleted or added to p to obtain p', only the order could be changed.

Now Polycarp asks you to help him to implement the password check module. Given the password p and the hash h, check that h can be the hash for the password p.

Your program should answer t independent test cases.

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains a non-empty string p, consisting of lowercase Latin letters. The length of p does not exceed 100.

The second line of each test case contains a non-empty string h, consisting of lowercase Latin letters. The length of h does not exceed 100.

Output

For each test case print the answer to it — "YES" if the given hash h could be obtained from the given password p or "NO" otherwise.

Example

input

```
5
abacaba
zyxaabcaabkjh
onetwothree
threetwoone
one
zzonneyy
one
none
twenty
ten
```

output

```
YES
YES
NO
YES
NO
```

Note

The first test case is explained in the statement.

In the second test case both s_1 and s_2 are empty and $p' =$ "threetwoone" is p shuffled.

In the third test case the hash could not be obtained from the password.

In the fourth test case $s_1 =$ "n", s_2 is empty and $p' =$ "one" is p shuffled (even thought it stayed the same).

In the fifth test case the hash could not be obtained from the password.
