Problem 4 – Hidden Password

Source filename: `hiddenpwd.(cpp|java)`
Input filename: `hiddenpwd.in`
Output filename: `hiddenpwd.out`

Insecure Inc. has decided to shift directions after a failed attempt at developing a new encryption standard. Their effort is a password system used to hide a password inside another string of characters we denote as a *message*. However, it is important that the message has a certain property relative to the hidden password.

Let us assume that we denote the uppercase alphabet characters of the password as $c_1 c_2 \dots c_p$ (although those characters need not be distinct). To be a valid message for the password, if you start from the beginning of the message and search for any character from the set $\{c_1, c_2, \dots, c_p\}$, it must be that c_1 is the first that you find. Subsequently, if you continue looking from that point of the message for any character from the set $\{c_1, c_2, \dots, c_p\}$, it must be that c_2 is the next that you find. Continuing in that manner, c_3 must be the next character from the set $\{c_1, c_2, \dots, c_p\}$, and so on until reaching c_p.

For example, if the password is `ABC`, then the string `H`**`A`**`PPY`**`B`**`IRTHDAY`**`C`**`ACEY` is a valid message.

- Notice that `A` is the first of the set `{A,B,C}` to appear in the message. (The initial `H` is not relevant.)
- Following the `A` that was found, the next occurrence from the set `{B,C}` is `B`.
- Following the `B` that was found, the next occurrence from the set `{C}` is indeed `C`. (Note that the `A` in `DAY` is not relevant, since we are only looking for a `C` at this point, and the additional `A` and `C` in `CACEY` are not relevant, because we have already completed the password with the first `C`.)

However for the password `ABC`, the string `TR`**`A`**`GICBIRTHDAYCACEY` is not a valid message.

- While the `A` is the first of the set `{A,B,C}` to appear in the string, the next occurrence from the set `{B,C}` is `C` rather than `B`.

Also, the string `H`**`A`**`PPY`**`B`**`IRTHDAY` is not a valid message for the password `ABC` because the `C` never appears.

As an example with duplicate letters in the password, consider the password `SECRET`. For this password, the string **`S`**`OM`**`EC`**`HO`**`RE`**`SARE`**`T`**`OUGH` is a valid message. In contrast, the string **`S`**`OM`**`EC`**`HEERSARETOUGH` is not a valid message, because then extraneous `E` is found at the point when an `R` is expected.

Input File (hiddenpwd.in)

The input file contains several test cases. The first line contains a positive integer that represents the number of test cases that follow. Each test case consists of a single line that contains 2 strings separated by at least 1 space. The first string is the password, having length P ($where\ 3 \leq P \leq 8$). The second string has length S ($where\ 10 \leq S \leq 40$). Both strings will consist solely of uppercase letters. (That is, neither string can include whitespace, lowercase letters, digits, or other special characters.)

Output File (hiddenpwd.out)

For each test case, output a single line with the word `PASS` if the second string is a valid message for the password. Otherwise output the word `FAIL`.

Example Input	***Example Output***
5 ABC HAPPYBIRTHDAYCACEY ABC TRAGICBIRTHDAYCACEY ABC HAPPYBIRTHDAY SECRET SOMECHORESARETOUGH SECRET SOMECHEERSARETOUGH	PASS FAIL FAIL PASS FAIL