Problem D. DigiD

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

As a Dutch citizen, you and your partner Daan have a DigiD account to do your taxes online. Daan has shared his password with you, and you have pointed out to him that his password is not very strong, so he should change his password to a new, stronger password. He asked you for advice on what makes a good password, and you have decided to prank him a little for his lack of cryptographic knowledge by giving him a ridiculous list of rules that his password should satisfy in order to be secure. You have settled on sharing the following list of ten rules:

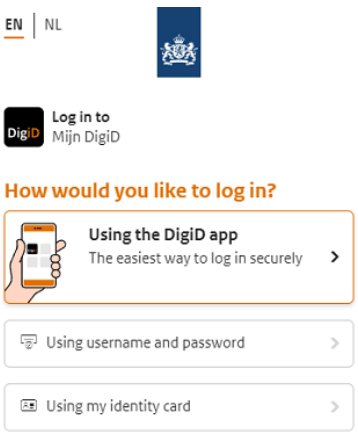


Figure 4: The log-in screen for DigiD.

In the following rules, note the difference between **substring** and **subsequence**. [†]

1. Only non-whitespace printable ASCII characters are used (ASCII codes 33 through 126).
2. At least one character is an uppercase English letter.
3. At least one character is a special ASCII character: not an English letter or a numerical digit.
4. At least one character is a lowercase English letter.
5. There needs to be a palindromic substring of length at least 3.
6. The subsequence formed by all the round bracket characters `()` must form a valid bracket sequence. A valid bracket sequence is defined recursively as either an empty sequence, the concatenation of two valid bracket sequences, or an opening bracket `(` followed by a valid bracket sequence followed by a closing bracket `)`.
7. There must not be a substring of 5 consecutive identical numerical digits.
8. The subsequence formed by all the numerical digit characters must not be monotone: there must be a strict increase somewhere, and there must be a strict decrease somewhere.
9. At least two distinct elements from the periodic table with a two-letter code consisting of one uppercase English letter followed by one lowercase English letter must occur as substrings in the password. In the cheatsheet in the attachments, all the allowed two-letter elements are given.
10. We consider a greedy algorithm for extracting Roman numerals from the password. Each time, take the leftmost character that is one of the uppercase letters `IVXLCDM`. Then, find the longest substring starting at this position that describes a Roman numeral in standard form, and add it to the list of Roman numerals (such a substring is guaranteed to exist, because the character itself is already a Roman numeral). Finally, you delete this substring and repeat until no character from `IVXLCDM` remains. Then the product of these Roman numerals must be 69, and all of the terms must be at least 2.

A complete list of two-letter elements from the periodic table and standard form Roman numerals is in the attached file `cheatsheet.md`.

[†]A substring is a contiguous sequence of characters in the string. A subsequence is obtained by deleting some (or none) of the characters from the string, and concatenating the remaining characters in their original order.

Daan is obviously overwhelmed with your insane list of rules. "Prank succeeded! HAHA!" You think to yourself, but your prank will soon backfire. Daan is very worried that he can not come up with a password that obeys all ten rules, but he will also be able to remember himself. He figured that it should be doable to remember his new password if it is an anagram of his old password, but he struggles to find a permutation that obeys all ten rules. He has given up hope of finding such a permutation, which is a big problem, because he does need to update his password, so you need to help him. You don't want to admit that this was all a prank, because that might diminish the trust in your relationship with Daan. You have thus decided that your only option is to write a program that finds a proper new password for Daan: a permutation of his old password into an anagram that obeys all ten rules.

Input

One string s ($1 \leq |s| \leq 10^5$) describing Daan's old password. It is guaranteed that Daan's old password obeys the first of the ten rules.

Output

First, one string, either `YES` or `NO`, indicating whether a proper new password for Daan is possible. If the first output string is `YES`, then follow it up with another string, describing such a proper new password for Daan.

Examples

standard input	*standard output*
`xX_ILoveSaskia2026_Xx`	`YES` `2026LXIXeSi__aakosvxx`
`password12345`	`NO`
`1234!@#$)(BHIIIIIIXXee`	`YES` `4312!#$()@BeHeIIIXXIII`
`Neapolis2010!`	`NO`