Overview: Among all substrings longer than a given length, and among all the anagrams of these substrings, find the most common one.

Description: Yay! Thanks to the most recent scuffle with evil, Hulk has been able to do what he likes: smash. Unfortunately, in his great enthusiasm, he's smashed a warehouse (represented by a string) containing important items that S.H.I.E.L.D owns. Okay, maybe he was forcibly ejected off a plane and didn't actually *choose* to smash into the warehouse, but details, details.

Anyhow, the items are probably beyond repair, but S.H.I.E.L.D. would like to figure out an approximation of what the most well-stocked item in the warehouse was. Items are represented as substrings of length *n* or more, and each substring has been smashed into an anagram. A substring is a contiguous sequence of characters inside of a given string. For example, the string "abc" has substrings "a", "b", "c", "ab", "bc", and "abc". An anagram of a string is some rearrangement of the letters. For example, the anagrams of "abc" are "abc", "acb", "bac", "bca", "cab", and "cba".

Given a warehouse (a string) and a length *n*, consider all substrings of the string with length greater than or equal to *n*. These are all the currently smashed items in the warehouse. And for all these substrings, consider all of their anagrams - these are all the un-smashed items that a smashed item could once have been. Output the substring anagram (i.e. item) that occurs the most frequently; if there are multiple, output the lexicographically smallest one (i.e. the one that would come first when sorted alphabetically).

Filename: nov91.{java, cpp, c, cc, py}

Input: The input file will consist of exactly two lines. The first line will contain the given string. The second line will give the length *n* - the minimum length of substrings to consider.

Output: The most common anagram among all substrings with length greater than or equal to *n*, choosing the lexicographically smallest among ties.

Assumptions: The given string will be no longer than 1000 characters.
n will be less than or equal to the length of the string.

Sample Input #1:
```
bbbabbbaaabaaabbb
4
```

Sample Output #1:
```
aaab
```

Sample Input #2:
```
abcdefghijklmnopqrstuvwxyz
25
```

Sample Output #2:
```
abcdefghijklmnopqrstuvwxy
```