As is well known, an alphabet is a standardized set of letters, and a word is the smallest free form in a language; it can be written as a sequence of letters and symbolizes a meaning. Letters, as elements of alphabets, have a prescribed order, generally known as *alphabetical order*. The principle behind extending the alphabetical order to words (*lexicographical order*) is that all words in a list beginning with the same letter should be grouped together, and before any words starting with a letter that comes later in alphabetical order; within a group of all words starting with the same letter, all words beginning with the same two letters shall be grouped together, and so on; thus, when comparing two words for lexicographical order, the ordering is determined by the alphabetical order of the two letters at the position where the two words first differ. If a word is a prefix of another, the former comes before the latter.

Here we are interested in a generic kind of words, bearing no relation with any specific language. Each of such, let's say, *pseudo-words* will be an ordered subset of letters (that is to say, no letter is repeated and it does not matter whether the word has a meaning in any actual language or not). As the set of available letters we will use the lower case (also called minuscule) form of the Latin alphabet, along with their standard alphabetical order:

`a<b<c<d<e<f<g<h<i<j<k<l<m<n<o<p<q<r<s<t<u<v<w<x<y<z`

Your task is to calculate the position of a given string (its *rank*) in the list of all the pseudo-words we can generate by using only characters of the string (remember that all of them are different), sorted in lexicographical order, as well as to find the pseudo-words corresponding to one or more given ranks.

Input

The input consists of several test cases. The first line, for each of them, contains a string of (distinct) characters of the Latin alphabet in lower case ('`a`' - '`z`'). The length of such a string will be between 1 and 20, inclusive. The following lines will contain an integer between 1 and the total number of pseudo-words that is possible to form with the letters of the string, until a new string is found or the input file ends.

Output

For each test case, output a line with the rank of the input string in the list of all pseudo-words, and then print in a line by itself each of the pseudo-words ranked at the positions of the corresponding input numbers.

Sample Input

```
cadb
1
64
38
13
23
abcdefghijklmnopqrst
1
21
```

Sample Output

```
38
a
dcba
cadb
adb
bc
20
a
abcdefghijklmnopqrt
```