Zig-Zag Permutation

In the following we will deal with nonempty words consists only of lower case letters 'a','b',..., 'j' and we will use the natural 'a' < 'b' < ... < 'j' ordering. Your task is to write a program that generates almost all zig-zag words (zig-zag permutations) from a given collection of letters. We say that a word W=W(1)W(2)...W(n) is zig-zag iff n = 1 or W(i) > W(i+1) and W(j) < W(j+1) for all odd 0 < i < n and for all even 0 < j < n or W(i) > W(i+1) and W(j) < W(j+1) for all even 0 < i < n and for all odd 0 < j < n. For example: "aabcc" is not zig-zag, "acacb" is zig-zag, "cac" is zig-zag, "abababc" is not zig-zag. If you imagine all possible zig-zag permutations of a word in increasing lexicographic order, you can assign a serial number (rank) to each one. For example: the word "aabcc" generates the sequence: 1 <-> "acacb", 2 <-> "acbca", 3 <-> "bacac", 4 <-> "bcaca", 5 <-> "cabac", 6 <-> "cacab".

Input

The input file consists several test cases. Each case contains a word (W) not longer than 64 letters and one positive number (D). The letters of each word are in increasing order. Input terminated by EOF.

Output

For each case in the input file, the output file must contain all of the zig-zag permutations of W whose zig-zag serial is divisible by D, in increasing lexicographic order - one word per line. In the next line you have to print the total number of zig-zag permutations of W. There is no case that produces more than 365 lines of output. Print an empty line after each case.

Example

Input:
j 1
abc 2
aaabc 1
aaabb 2
aaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbcccdd 123456

Output:
j
1

bac
cab
4

abaca
acaba
2

1

babacbcabacabadababababababababababadab
213216