A string S is given. In one move, any two adjacent letters can be swapped. For example, given a string "abcd", it's possible to create "bacd", "acbd" or "abdc" in one such move. What is the lexicographically minimum string that can be achieved by at most K moves?

Write a function:

def solution(S, K)

that, given a string S of length N and an integer K, returns the lexicographically minimum string that can be achieved by at most K swaps of any adjacent letters.

Examples:

1. Given S = "decade" and K = 4, your function should return "adcede". Swaps could be:

decade → dceade,

dceade → dcaede,

dcaede → dacede,

dacede → adcede.

2. Given S = "bbbabbb" and K = 2, your function should return "babbbbb". The swaps are:

bbbabbb → bbabbbb,

bbabbbb → babbbbb.

3. Given S = "abracadabra" and K = 15, your function should return "aaaaabrcdbr".

Write an efficient algorithm for the following assumptions:

N is an integer within the range [1..100,000];

string S consists only of lowercase letters ('a'-'z');

K is an integer within the range [0..1,000,000,000].