Given a string, construct a new string by rearranging the original string and deleting characters as needed. Return the alphabetically largest string that can be constructed respecting a limit as to how many consecutive characters can be the same.

****Example****

*s = 'baccc'*

*k = 2*

The largest string, alphabetically, is '*cccba'* but it is not allowed because it uses the character '*c*' more than 2 times consecutively. Therefore, the answer is '*ccbca'*.

**Function Description**

Complete the function *getLargestString* in the editor below.

getLargestString has the following parameters:

    string *s[n]:*  the original string

    int *k:* the maximum number of identical consecutive characters the new string can have

Returns:

*string*: the alphabetically largest string that can be constructed that has no more than *k* identical consecutive characters

**Constraints**

1 ≤ *n* ≤ 105

1 ≤ *k* ≤ 103

The string *s* contains only lowercase English letters.

Input Format For Custom Testing

The first line contains a string, *s*.

The second line contains an integer, *k*.

Sample Case 0

**Sample Input**

STDIN Function

-----     -----

zzzazz *→* string s = 'zzzazz'

2 → k = 2

**Sample Output**

zzazz

**Explanation**

One '*z*' must be removed so that no more than 2 consecutive characters are the same.

Sample Case 1

**Sample Input**

STDIN Function

----- -----

axxzzx → s = 'axxzzx'

2 → k = 2

**Sample Output**

zzxxax

**Explanation**

The character '*a*' must separate the 3 '*x*' characters so that no more than 2 consecutive characters are the same.

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

class Result {

/\*

\* Complete the 'getLargestString' function below.

\*

\* The function is expected to return a STRING.

\* The function accepts following parameters:

\* 1. STRING s

\* 2. INTEGER k

\*/

public static String getLargestString(String s, int k) {

// Write your code here

}

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

String s = bufferedReader.readLine();

int k = Integer.parseInt(bufferedReader.readLine().trim());

String result = Result.getLargestString(s, k);

bufferedWriter.write(result);

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*