Given a string you need to print the size of the longest possible substring that has exactly K unique characters. If there is no possible substring then print -1.

Input Format

Take the String S as in input

Take the Integer K as in input

Constraints

1 ≤ |S| ≤ 10^5

1 ≤ K ≤ 26

Output Format

returns the length of the longest substring with exactly K distinct characters. If there is no substring with exactly K distinct characters then return -1.

Sample Input 0

```
aabacbebebe
3
```

Sample Output 0

```
7
```

Sample Input 1

```
aaaa
2
```

Sample Output 1

```
-1
```

Language: Java 8

1st approach

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s1=sc.nextLine();
        String s2=sc.nextLine();


        HashMap<Character,Integer> hm=new HashMap<>();
        for(int i=0;i<s1.length();i++){
            char ch =s1.charAt(i);
            hm.put(ch,hm.getOrDefault(ch,0)+1);
        }
        int l=s1.length();
        for(int i=0;i<=s2.length()-l;i++){

            String str= s2.substring(i,i+l);
              HashMap<Character,Integer> hm1=new HashMap<>();
        for(int j=0;j<str.length();j++){
            char ch =str.charAt(j);
            hm1.put(ch,hm1.getOrDefault(ch,0)+1);
        }

            if(hm.equals(hm1)){System.out.print("true");return;}

        }

        System.out.println("false");
    }
}
```

2nd approach

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Y
        Scanner sc = new Scanner(System.in);
        String s1 = sc.next();
        String s2 = sc.next();
        System.out.println(checkInclusion(s1, s2));
    }
    public static boolean checkInclusion(String s1, String s2){
        if(s1.length() > s2.length()){
            return false;
        }
        int[] s1Count = new int[26];
        int[] s2Count = new int[26];
        for(char c : s1.toCharArray()){
            s1Count[c-'a']++;
        }
        int windowSize = s1.length();
        for(int i = 0; i < s2.length(); i++){
            s2Count[s2.charAt(i)-'a']++;
            if(i >= windowSize){
                s2Count[s2.charAt(i-windowSize)-'a']--;
            }
            if(Arrays.equals(s1Count, s2Count)){
                return true;
            }
        }
        return false;
    }
}
```

HW_common prefix 1

locked

Problem | Submissions | Leaderboard | Discussions

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

Input Format

a number N representing length of array

N number of string inputs as element of array.

Constraints

1 <= strs.length <= 200 0 <= strs[i].length <= 200 strs[i] consists of only lowercase English letters.

Output Format

print the longest common prefix.

Sample Input 0

```
3
flower flow flight
```

Sample Output 0

```
fl
```

Code

Java | Auto

```java
class Solution {
    public String longestCommonPrefix(String[] strs) {
      int n = strs.length;
        int m = Integer.MAX_VALUE;
        for(int i = 0; i < n; i++){
            m = Math.min(m,strs[i].length());
        }

        String ans = "";
        for(int i = 0; i < m; i++){
            char ch = strs[0].charAt(i);
            boolean bl = true;
            for(int j = 1; j < n; j++){
                if(ch != strs[j].charAt(i)){
                    bl = false;
                    return ans;

                }
            }
            if(bl)
            ans = ans + ch;
        }
        return ans;
    }
}
```

Equally Frequent Letters

locked

Problem | Submissions | Leaderboard | Discussions

Given a string str consisting of lowercase English letters indexed from 0, you must choose one index and delete the corresponding letter. This deletion should result in every letter appearing with the same frequency in the remaining string.

If it's possible to remove one letter from String to satisfy the aforementioned condition, return true; otherwise, return false

NOTE:-

The frequency of a letter x is the number of times it occurs in the string.

You must remove exactly one letter and cannot chose to do nothing.

Input Format

take String str as input.

Constraints

2 <= word.length <= 10^4

word consists of lowercase English letters only.

Output Format

return true; otherwise, return false

Sample Input 0

```
abcc
```

Sample Output 0

```
true
```

Sample Input 1

```
aazz
```

Sample Output 1

```
false
```