package com.iimtiaz.day\_08;  
  
public class PermutationInString {  
 public static void main(String[] args) {  
 String s1 = "ab", s2 = "eidbaooo";  
 System.*out*.println(new Solution\_3().checkInclusion(s1, s2));  
 }  
}  
  
  
*/\*\*  
 \* Time Complexity: O(m + n)  
 \* O(n): Initializing the count array and processing the first n characters of both strings.  
 \* O(m - n): Iterating through the remaining characters of s2 (if any) to check for matching permutations.  
 \* O(1): allZeros function calls are constant time, as they involve short loops of fixed length (26).  
 \* Space Complexity: O(1)  
 \* Uses a fixed-size array count to store character frequencies, regardless of input string lengths.  
 \* No other data structures with size dependent on input are used.  
 \*/*class Solution\_3 {  
 public boolean checkInclusion(String s1, String s2) {  
 int n = s1.length();  
 int m = s2.length();  
 if (n > m) return false;  
 int[] count = new int[26];  
 for (int i = 0; i < n; i++) {  
 count[s1.charAt(i) - 'a']++;  
 count[s2.charAt(i) - 'a']--;  
 }  
 if (allZeros(count)) return true;  
  
 for (int i = n; i < m; i++) {  
 count[s2.charAt(i) - 'a']--;  
 count[s2.charAt(i - n) - 'a']++;  
 if (allZeros(count)) return true;  
 }  
 return false;  
 }  
  
 private boolean allZeros(int[] count) {  
 for (int i = 0; i < 26; i++) {  
 if (count[i] != 0) return false;  
 }  
 return true;  
 }  
}