Find if string is K-Palindrome or not

Given a string, find out if the string is K-Palindrome or not. A k-palindrome string transforms into a palindrome on removing at most k characters from it.

Examples :

```
Input : String - abcdecba, k = 1

Output : Yes

String can become palindrome by remo-

-ving 1 character i.e. either d or e)



Input  : String - abcdeca, K = 2

Output : Yes

Can become palindrome by removing

2 characters b and e.


Input : String - acdcb, K = 1

Output : No

String can not become palindrome by

removing only one character.
```

We strongly recommend you to minimize your browser and try this yourself first.

If we carefully analyze the problem, the task is to transform the given string into its reverse by removing at most K characters from it. The problem is basically a variation of Edit Distance. We can modify the Edit Distance problem to consider given string and its reverse as input and only operation allowed is deletion. Since given string is compared with its reverse, we will do at most N deletions from first string and N deletions from second string to make them equal. Therefore, for a string to be k-palindrome, 2*N <= 2*K should hold true. Below are the detailed steps of algorithm - Process all characters one by one staring from either from left or right sides of both strings. Let us traverse from the right corner, there are two possibilities for every pair of character being traversed.

1. If last characters of two strings are same, we ignore last characters and get count for remaining strings. So we recur for lengths m-1 and n-1 where m is length of str1 and n is length of str2.
2. If last characters are not same, we consider remove operation on last character of first string and last character of second string, recursively compute minimum cost for the operations and take minimum of two values.
    - Remove last char from str1: Recur for m-1 and n.
    - Remove last char from str2: Recur for m and n-1.

Below is Naive recursive C++ implementation of above approach.

```
// A Naive recursive C++ program to find
// if given string is K-Palindrome or not
#include<bits/stdc++.h>
using namespace std;
```

```
// find if given string is K-Palindrome or not
int isKPalRec(string str1, string str2, int m, int n)
{
    // If first string is empty, the only option is to
    // remove all characters of second string
    if (m == 0) return n;

    // If second string is empty, the only option is to
    // remove all characters of first string
    if (n == 0) return m;

    // If last characters of two strings are same, ignore
    // last characters and get count for remaining strings.
    if (str1[m-1] == str2[n-1])
        return isKPalRec(str1, str2, m-1, n-1);

    // If last characters are not same,
    // 1. Remove last char from str1 and recur for m-1 and n
    // 2. Remove last char from str2 and recur for m and n-1
    // Take minimum of above two operations
    return 1 + min(isKPalRec(str1, str2, m-1, n), // Remove from
str1
                   isKPalRec(str1, str2, m, n-1)); // Remove from
str2
}
// Returns true if str is k palindrome.
bool isKPal(string str, int k)
{
    string revStr = str;
    reverse(revStr.begin(), revStr.end());
    int len = str.length();
     return (isKPalRec(str, revStr, len, len) <= k*2);
}

// Driver program
int main()
{
    string str = "acdcb";
    int k = 2;
    isKPal(str, k)? cout << "Yes" : cout << "No";
    return 0;
}
```