1. Reverse Word in Odd/Even Positions

You are expected to reverse words in a sentence either at odd or even position based on the input provided by the user.

Input : he likes apples
odd
Output : eh likes selppa

Input : we love to cook food using grill
even
Output : we evol to kooc food gnisu grill

2. String Processing

You are expected to process the input string (Case-sensitive), remove duplicate characters and display the occurrence count of the corresponding character in the entire string next to it.

Input : occurrence
Output : o1c3u1r2e2n1

Input : onechar
Output : o1n1e1c1h1a1r1

Input : pokemon
Output : p1o2k1e1m1n1

3. Palindrome Substring

You are expected to split String into 2 Substrings in a linear fashion and check whether the resulting SubStrings are Palindrome or not.

Input : civicbob
Output : Can be split into Palindrome substrings
Explanation : civic and bob are the two Palindrome Substrings

Input : tester
Output : Can't be split into Palindrome substrings
Explanation : No Palindrome Substrings can be obtained from tester.

4. Longest Substring without repetition of Characters

Given a string s, find the length of the longest substring without repeating characters.

Input: abcabcbb
Output: 3
Explanation: The answer is "abc", with the length of 3.

Input: bbbbb
Output: 1
Explanation: The answer is "b", with the length of 1.

Input: pwwkew
Output: 3
Explanation: The answer is "wke", with the length of 3.

Input:
Output: 0

5. Write a program that takes two arguments at the command line, both st
The program checks to see whether or not the second string is a substring
first (without using the substr or any other library function). One caveat: any
the second string can match zero or more characters in the first string, so
input were abcd and the substring were a*c, then it would count as a subs
Also, include functionality to allow an asterisk to be taken literally if prece
a '\', and a '\' is taken literally except when preceding an asterisk.

Examples :

Input : abcd , a*c	**Output** : true
Input : spoon , sp*n	**Output** : true
Input : regex , re*g	**Output** : true
Input : search , *c	**Output** : true
Input : zoho , *o*o	**Output** : true
Input : zoho , *ogo	**Output** : false
Input : test , pest	**Output** : false
Input : st*r , t*r	**Output** : true
Input : star , t*r	**Output** : false
Input : tree , tr\	**Output** : false
Input : tr\e , tr\	**Output** : true

6. Smallest distance between any two occurrences of word0 and word1

Given the strings text, word0, and word1, return the smallest distance between any two occurrences of word0 and word1 in text in any order, measured in number of words
in between.
The text can have punctuation after a word that's not considered a part of that word. If either word0 or word1 doesn't appear in text, return -1.

Input Format:
The first line of the input consists of one string.
The second line of the input consists of word0.
The third line of the input consists of word1.

Input:

```
hi there, how have you been?
hi
hi
```

Output:

```
-1
```

Input:

```
hi there, how have you been?
there
been
```

Output:

```
3
```

Input:

```
hi there, hi here, there's a hi everywhere
hi
hi
```

Output:

```
1
```