String assignment questions

GitHub location to update the solution:

https://github.com/dracha-sf/j2e-fy24

1. Please clone the above repo.
2. Create a branch as student/<collegeName>/studentName
    a. For example student/bansasthali/priyanka

PROBLEM 1:

Two strings are said to be the same if they are of the same length and have the same character at each index. Backspacing in a string removes the previous character in the string.

Given two strings containing lowercase English letters and the character '#' which represents a backspace key, determine if the two final strings are equal. Return 1 if they are equal or 0 if they are not.
Note that backspacing an empty string results in an empty string.

Example:

```
s1 = 'axx#bb#c'
s2 = 'axbd#c#c'

In the first string, one 'x' and one 'b' are backspaced over.
The first string becomes axbc. The second string also becomes axbc.
The answer is 1.
```

PROBLEM 2:

Data, in the form of a binary string has to be sent across two servers. However, according to a new network control protocol, data can only be sent in the form of binary strings that have no two adjacent characters same. Such binary strings with no two adjacent characters same are called special strings. Any data to be transmitted is first broken into one/numerous subsequences that are special strings and then each special string is sent as a data packet across the connected servers.

Given a binary string that has to be sent across two servers, find the minimum number of data packets it will be broken into.
Note: A subsequence of a string is obtained by deleting some characters from the string while maintaining the order. For example, "011" is a subsequence of "0101" while "100" is not.

Example:

```
Suppose input_str = "00100"

The given string can be broken into three subsequences that are special as follows:
"0", "010", and "0"
It is also the minimum number of special subsequences that the string can be broken int
Hence the output is 3.
```

PROBLEM 3:

An English lecture at Elementary School is aimed at teaching students the letters of the alphabet.

The students are provided with a string *word* that consists of lowercase English letters. In one move, they can choose any index *i,* and let the character at that index be *c.* Then, the first occurrence of *c* to the left of *i,* and the first occurrence of *c* to the right of *i* are deleted (Note: the operation can still be carried out even if either the left or right occurrence doesn't exist).

For example, if *word = "adabacaea",* and if index 4 is chosen (0-based), the first occurrence of 'a' to the left and right of index 4 (bold, indices 2 and 6) are deleted leaving *word = "adbacea".*

Find the minimum number of moves the students need to perform in order to get a word of minimal length.

Example:

```
Consider word = "baabacaa".

The following moves are optimal.

1. Choose index 0, "baabacaa", then word = "baaacaa".
Delete the b to its right at index 3.
There is no b to its left so the operation is finished.
2. Now, choose index 2, "baaacaa", then word = "bacaa".
3. Now. choose index 3, "bacaa", then word = "bca".


The word cannot be reduced further. The answer is 3.
```

PROBLEM 4:

A string is a pangram if it contains all letters of the English alphabet, ascii['a'-'z']. Given a list of strings, determine if each one is a pangram or not. Return "1" if true and "0" if false.

Constraints

- *1 ≤ n ≤ 100*
- Each string *pangram[i]* (where *0 ≤ i < n*) is composed of *lowercase letters* and *spaces*.
- *1 ≤ length of pangram[i] ≤ 10^5*

Example:

```
pangram = ['pack my box with five dozen liquor jugs', 'this is not a pangram']

the string 'pack my box with five dozen liquor jugs' is a pangram ,
because it contains all the letters 'a' through 'z'

the string 'this is not a pangram' is not a pangram

Assemble a string of the two results, in order. The result is '10'.
```

PROBLEM 5:

Parentheses strings are strings containing only the characters '(' and ')'. A parentheses string is considered balanced when its opening parentheses align with its closing parentheses. For example, "()()" and "((()))" are balanced parentheses strings while

")(", "())(" are not.

Given a string consisting of the same number of opening and closing parentheses, determine the minimum number of character swaps required to make the string a balanced parentheses string.

Example:

```
s = "))(("

Swap the first and the last characters to get "()()", a balanced parentheses string.
The minimum number of swaps is 1.
```

PROBLEM 6:

The 26 characters of the alphabet are each assigned a security value represented as an array of integers, where *security_values[i]* is associated with the i^{th} character of the alphabet. Given an encrypted message, *msg,* and the array *security_values,* rearrange the characters in *msg* and find the minimum possible sum of the absolute differences of the security values of adjacent characters.

Example:

Given *security_values[i]* = [1, 2, 1, 3, 1, 3, 5, 7, 1, 1, 5, 5, 8, 10, 11, 1, 23, 2, 3, 7, 8, 9, 1, 6, 5, 9] and *s = "afeb".*
Here 'a' relates to 1, 'f' relates to 3, 'e' relates to 1, and 'b' relates to 2.
Some of the rearrangements are:

	A	B
1	**Rearrangement**	**Adjacent Difference**
2	afeb	|1 - 3| + |3 - 1| + |1 - 2| = 5
3	fbae	|3 - 2| + |2 - 1| + |1 - 1| = 2
4	eabf	|1 - 1| + |1 - 2| + |2 - 3| = 2

The optimal rearrangement is "eabf" or "fbae" with a sum of 2.

PROBLEM 7:

Given a string, how many different substrings exist in it that have no repeating characters? Two substrings are considered different if they have a different start or end index.

Constraints:

$1 \leq$ length of $s \leq 10^5$
s consists of only lowercase English letters, *ascii['a'-'z']*.

Example:

```
s = "abac"

The substrings that have no repeating characters in them are "a", "b", "a", "c", "ab",
Note that "aba" and "abac" do not qualify because the character 'a' is repeated in the
There are 8 substrings that have no repeating characters.
```

PROBLEM 8:

A palindrome is a string that reads the same from the left and from the right. For example, *mom* and *tacocat* are palindromes, as are any single-character strings. Given a string, determine the number of its substrings that are palindromes.

Constraints:

- $1 \leq |s| \leq 5 \times 10^3$
- *each character of s, s[i]* $\in$ *{'a'-'z'}.*

Example:

```
The string is s = 'tacocat'.
Palindromic substrings are ['t', 'a', 'c', 'o', 'c', 'a', 't', 'coc', 'acoca', 'tacoca
There are 10 palindromic substrings.
```

PROBLEM 9:

Different compression techniques are used in order to reduce the size of the messages sent over the web. An algorithm is designed to compress a given string by describing the total number of consecutive occurrences of each character next to it.

Constraints:

- *message[i]* $\in$ *ascii[a-z]*
- $|message| \leq 10^5$

Example:

```
For example, consider the string "abaasass".
Group the consecutive occurrence of each character:
'a' occurs one time.
'b' occurs one time.
'a' occurs two times consecutively.
's' occurs one time.
'a' occurs one time.
's' occurs two times consecutively.

If a character only occurs once, it is added to the compressed string.
If it occurs consecutive times, the character is added to the string followed by an in
Thus the compressed form of the string is "aba2sas2".
```

PROBLEM 10:

Given *word*, return the *next* alphabetically greater string in all permutations of that word. If there is no greater permutation, return the string *'no answer'* instead.

Constraints:

- *message[i]* $\in$ *ascii[a-z]*

- $|message| \leq 10^5$

Example:

```
word = 'baca'

The string 'baca' has the following permutations in alphabetical order:
'aabc', 'aacb', 'abac', 'abca', 'acab', 'acba', 'baac', 'baca', 'bcaa', 'caab', 'caba'
The next alphabetically greater permutation of the original string is 'bcaa'.
```